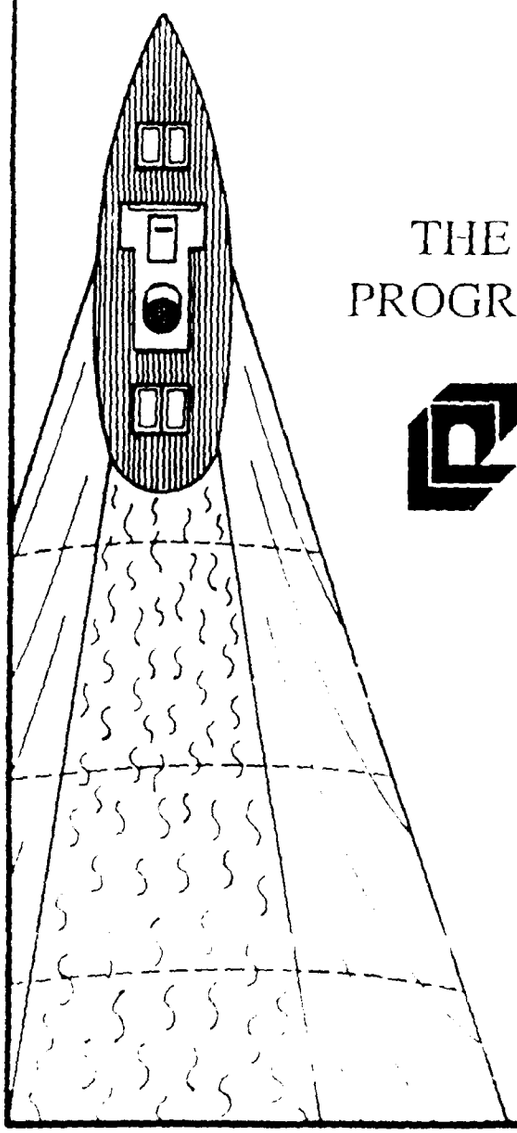


THE UNIVERSITY OF MICHIGAN
PROGRAM IN SHIP HYDRODYNAMICS

COLLEGE OF ENGINEERING

NAVAL ARCHITECTURE &
MARINE ENGINEERING

AEROSPACE ENGINEERING

MECHANICAL ENGINEERING &
APPLIED MECHANICS

SHIP HYDRODYNAMIC
LABORATORY

SPACE PHYSICS RESEARCH
LABORATORY

ERIM

(1)

Iso-Surface Construction

Klaus-Peter Beier and Soon-Hung Han
Department of Naval Architecture and Marine Engineering

Contract Number N00014-86-K-0684
Technical Report No. 90-3

June, 1990

ISO-SURFACE CONSTRUCTION

Soon-Hung Han
Klaus-Peter Beier

June 1990

The work described in this report was supported under the Program for Ship Hydrodynamics at The University of Michigan, funded by the University Research Initiative of the Office of Naval Research, Contract No. N000184-86-K-0684.

DEPARTMENT OF NAVAL ARCHITECTURE AND MARINE ENGINEERING
COLLEGE OF ENGINEERING THE UNIVERSITY OF MICHIGAN
ANN ARBOR, MICHIGAN 48109, U.S.A.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

This report describes the details of an algorithm which allows to find an iso-surface for a given three-dimensional property distribution. Input to the algorithm may be a set of scalar values given at regularly spaced grid points. Output will be triangulated polygonal approximation of the iso-surface for a specified property value.

Chapter 2

THE CONCEPT OF ISO-SURFACES

The concept of iso-surfaces assumes that a scalar property p (for example: pressure, temperature, or magnitude of velocity) is distributed inside a three-dimensional volume where it is continuously defined as $p = p(x, y, z)$. For our considerations, we assume that the volume of interest is a cuboid (a right parallelepiped) and bounded by rectangular faces. Therefore, the property distribution

$$p = p(x, y, z)$$

is defined for the given ranges of x, y, and z :

$$\begin{aligned} x_{min} &\leq x \leq x_{max}, \\ y_{min} &\leq y \leq y_{max}, \\ z_{min} &\leq z \leq z_{max}. \end{aligned}$$

Inside the volume, the scalar property p changes continuously from point to point and exhibits values within some problem dependent range

$$p_{min} \leq p \leq p_{max}.$$

To study the character of the three-dimensional property distribution, a constant property value p_c can be chosen and all points exhibiting this property value p_c can be found inside the given volume. The points of constant property value p_c can be determined from

$$p(x, y, z) - p_c = 0,$$

which is an implicitly defined three-dimensional surface. Such a surface is called an *iso-surface* and represents the locus of all points where the property p assumes the given constant property value p_c. The geometry of an iso-surface is problem dependent and varies from very simple shapes (for example: a plane, a sphere, or a cylinder) to extremely complex forms. Sample iso-surfaces are illustrated in Figure 2.1. As can be seen from Figure 2.1, a single iso-surface may consists of two or more unconnected surface parts.

Note that within a given property distribution, an infinite number of iso-surfaces can be found, each of them characterized by a different constant property value p_c. Figures 2.1a and 2.1b illustrate two iso-surfaces for two different p_c values within the same volume.

For a given p_c, an iso-surface exists inside a volume as long as

$$p_{min} \leq p_c \leq p_{max}.$$

If p_c is outside the range of p_{min} - p_{max}, then an iso-surface for the given p_c cannot be found within the given volume.

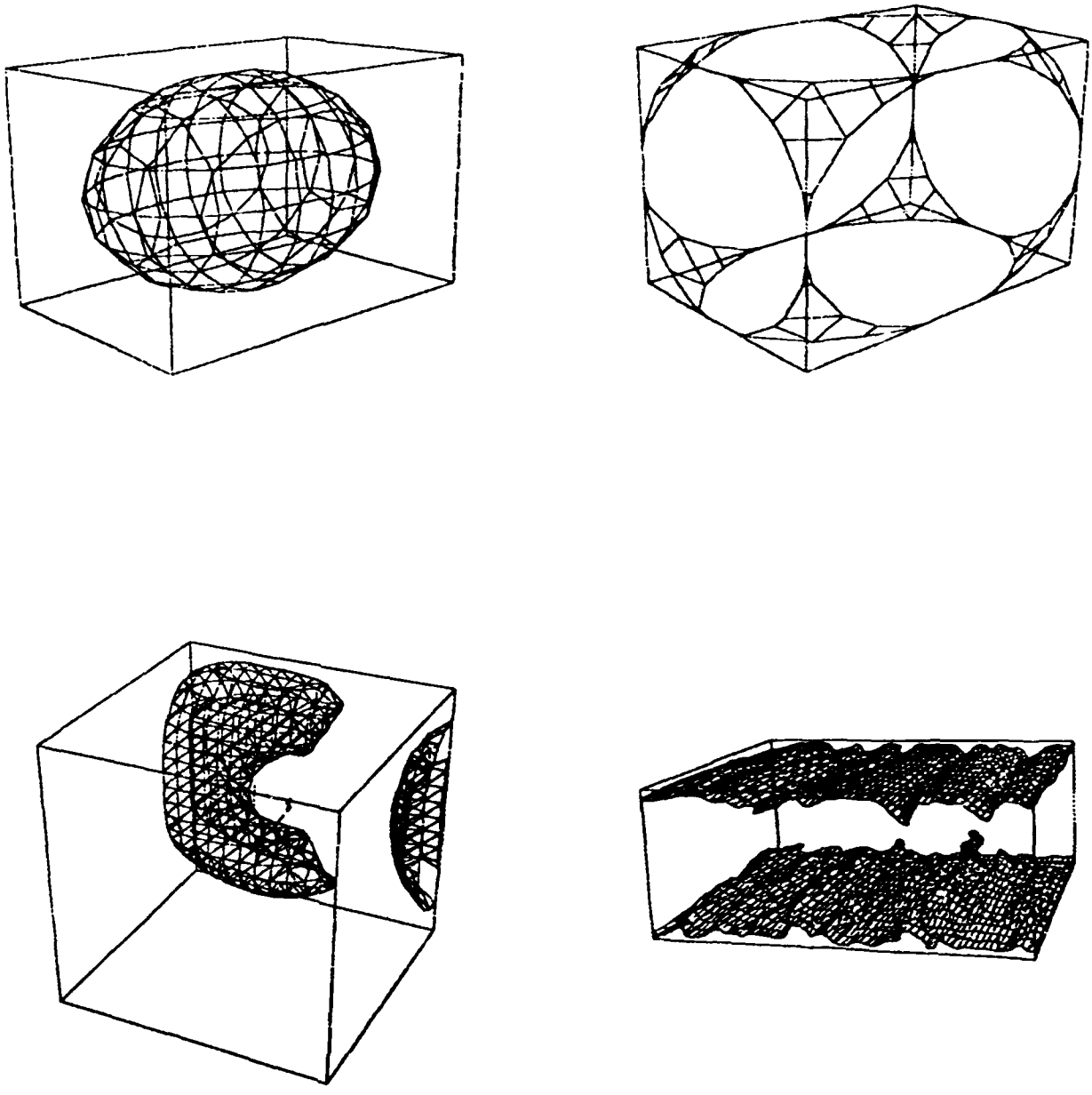


Figure 2.1: Examples of iso-surfaces

Chapter 3

PRINCIPLES OF ALGORITHM

3.1 Numerical Definition of Property Distribution

The algorithm which is described in the following requires a numerical definition of the property distribution $p = p(x, y, z)$ inside a given volume. Discrete property values $P_{i,j,k}$ must be provided at the intersection points of a rectangular, three-dimensional mesh of straight lines. Figure 3.1 illustrates a volume with a sample regular grid.

The grid is defined by the arrays

$$X_i, \quad i \text{ from 1 to } NX,$$

$$Y_j, \quad j \text{ from 1 to } NY,$$

$$Z_k, \quad k \text{ from 1 to } NZ.$$

Note that this definition allows for variable spacing between the grid lines in all three directions. The intersections of the grid lines are the grid points defined as the vectors

$$\mathrm{GP}_{i,j,k} = (X_i, Y_j, Z_k).$$

The total number of grid points NG for a given volume is

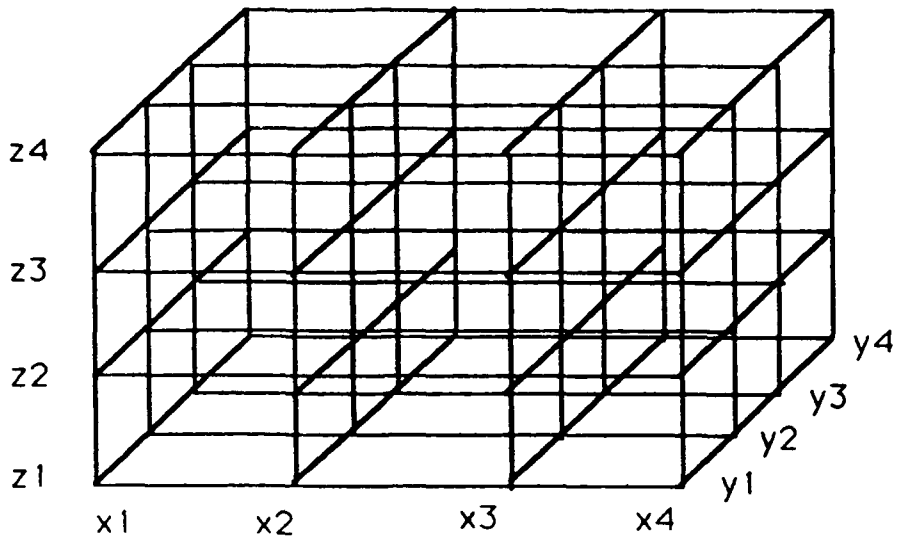


Figure 3.1: Computational volume with Grid lines

$$NG = NX \times NY \times NZ.$$

At each of the grid points $GP_{i,j,k}$, the scalar property value $P_{i,j,k}$ is given. The range of the property values for a given property distribution can be found from

$$P_{MIN} = MIN\ (P_{i,j,k},\ \ i\ = 1, NX; j = 1, NY; k = 1, NZ)$$

$$P_{MAX} = MAX\ (P_{i,j,k},\ \ i\ = 1, NX; j = 1, NY; k = 1, NZ)$$

3.2 Subdividing the Volume into Boxes

The regular grid can be used to divide the given volume into sub-volumes. Each sub-volume is again a cuboid or right parallelepiped and, for simplicity, will be called *box* or *grid box* in the following. A box extends between two consecutive grid lines in

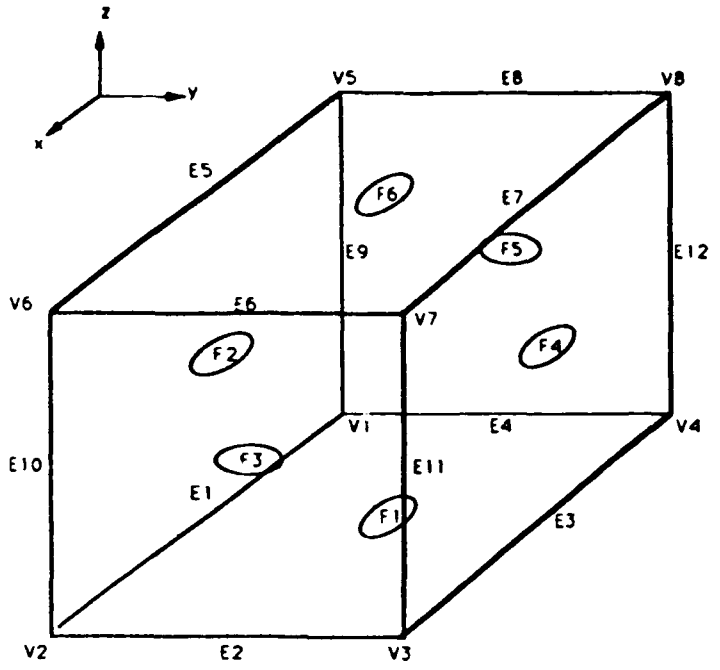


Figure 3.2: Local Identification System of a Box

all three directions, i.e., a box is bounded by the six planes $X = X_i$ and $X = X_{i+1}$, $Y = Y_j$ and $Y = Y_{j+1}$, and $Z = Z_k$ and $Z = Z_{k+1}$. The total number of boxes NB inside the given volume is

$$NB = (NX - 1) \times (NY - 1) \times (NZ - 1).$$

Figure 3.2 illustrates single box and introduces a local identification system for the geometric elements of a box:

- A box is defined by the 8 vertices $\mathbf{V}_k$, $k = 1,8$. The vertices coincide with the given grid points $\mathbf{GP}_{i,j,k}$. At each vertex $\mathbf{V}_k$, the property value P_k is known.
- The 8 vertices form the 12 edges E_l, $l = 1,12$.
- The 12 edges define 6 rectangular faces F_m, $m = 1,6$.

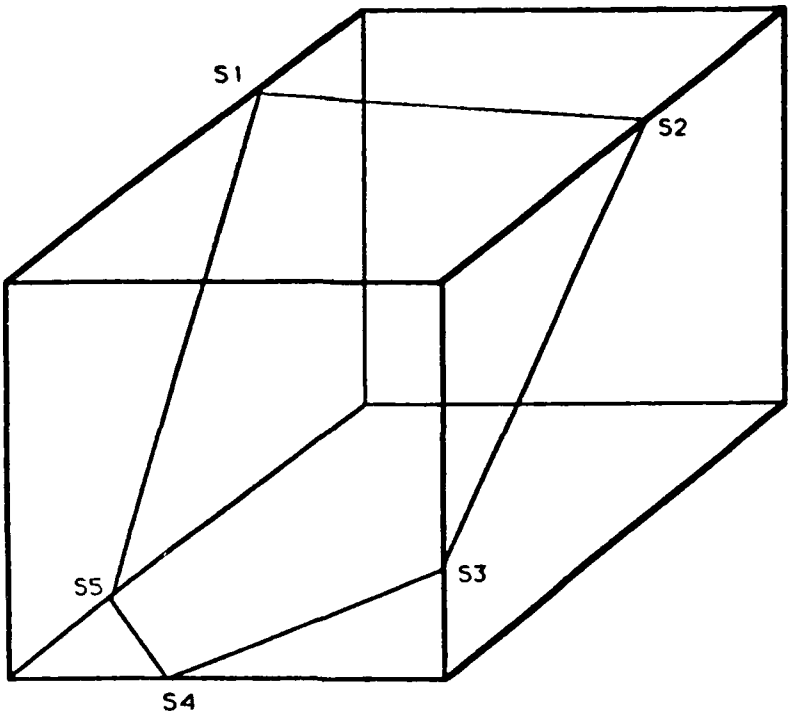


Figure 3.3: Iso-surface patch within a Box

3.3 Box-by-Box Surface Construction

To find the iso-surface for a given constant property value P_C, a box-by-box surface construction method is used. Each of the NB boxes is investigated individually. If the iso-surface intersects with single box, a patch of the iso-surface which lies inside the box can be found as illustrated in Figure 3.3. First, the intersection points $\mathbf{S}_i$ $(i = 1, NS)$ of the iso-surface with the edges of the box are found assuming a linear distribution of the property value along each edge. Next, the intersection points $\mathbf{S}_i$ are connected by line segments. Each line segment lies in one of the faces of the box. The line segments build the boundaries of the iso-surface patch inside the box and define a closed polygon which, in general, is non-planar. As shown later, the number of intersection points NS may vary between 3 and 12. Therefore, the shape of the resulting polygons may range from a simple triangle to a 12-sided polygon. In some

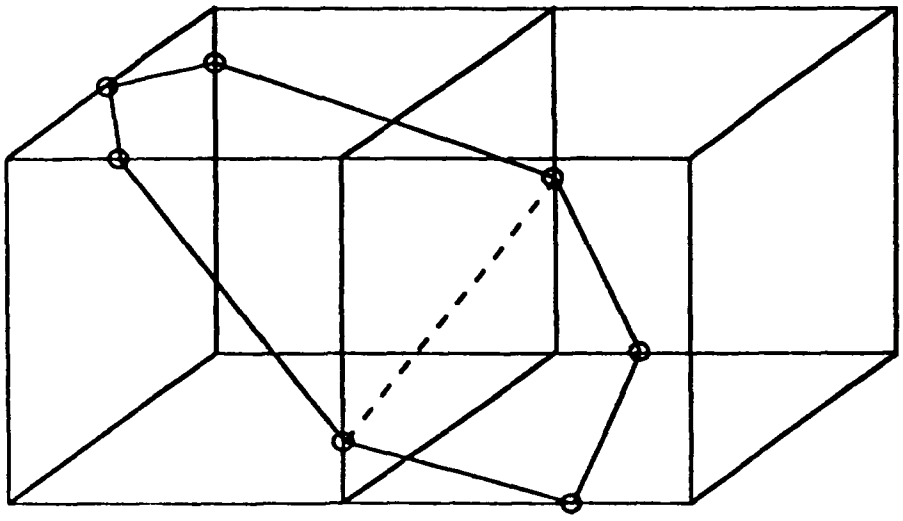


Figure 3.4: Two boxes sharing a face

cases, more than one polygon may result from the process. All resulting polygons can be decomposed into triangles. These triangles are used as the basic geometric elements for the iso-surface representation.

The entire iso-surface is constructed by combining the surface patches found within each of the NB boxes of the volume. If the iso-surface extends from one box into a neighboring box, the transition is always continuous: Both boxes share a common face and the polygons from both boxes share the same edge lying in that face. This is illustrated in Figure 3.4. Therefore, the sum of all polygons (or the sum of all triangles decomposed from the polygons) represent the entire iso-surface inside the given volume.

Not all of the NB boxes of the volume contribute to the construction of the

surface. An individual box is not intersected by the iso-surface if

$$P_C > P_k,\ k = 1,8$$

or if

$$P_C < P_k,\ k = 1,8.$$

Such a box is an *empty* box. Moreover, the entire given volume may be *empty* if

$$P_C > P_{MAX}$$

or

$$P_C < P_{MIN}.$$

In this case, no iso-surface exists inside the entire volume for the given value P_C.

3.4 Polygonal Approximation of Iso-surface Patch within a Box

Three levels of approximations are applied to the construction of iso-surface patch inside each grid box. First, the intersection points S_i are found using linear interpolation along each edge. Second, boundary curves of the iso-surface patch are approximated by straight lines. Finally, the iso-surface patch is approximated by sum of triangles.

The bilinear interpolation can be applied on a face of a box to show a part of above approximation process. This interpolation can be used to find out traces of the iso-surface on the face.

Bilinear interpolation is defined by four corner points[3]

$$\mathbf{P}(u,v) : \mathbf{P}(0,0), \mathbf{P}(0,1), \mathbf{P}(1,0), \text{ and } \mathbf{P}(1,1)$$

The corner points define linear boundary curves (straight lines) for the patch which is formed by the interpolation. Any points on the patch is linearly interpolated by

$$\mathbf{Q}(u,v) = \mathbf{P}(0,0)(1-u)(1-v) + \mathbf{P}(0,1)(1-u)v + \mathbf{P}(1,0)u(1-v) + \mathbf{P}(1,1)uv$$

For one face of a box we can use the more familiar x, y and z coordinates where z coordinate represents the property value P at a point defined by x and y. Also, the bilinear interpolation can be rewritten in matrix form

$$\mathbf{Q}(x,y) = \begin{bmatrix} (1-x)x \end{bmatrix} \begin{bmatrix} \mathbf{P}(0,0) & \mathbf{P}(0,1) \\ \mathbf{P}(1,0) & \mathbf{P}(1,1) \end{bmatrix} \begin{bmatrix} (1-y) \\ y \end{bmatrix}$$

or

$$\mathbf{Q}(x,y) = \begin{bmatrix} (1-x)(1-y) & (1-x)y & x(1-y) & xy \end{bmatrix} \begin{bmatrix} \mathbf{P}(0,0) \\ \mathbf{P}(0,1) \\ \mathbf{P}(1,0) \\ \mathbf{P}(1,1) \end{bmatrix}$$

Figure 3.5 shows one bilinear patch defined by four corner points

$$\begin{aligned} \mathbf{P}(0,0) &= (0,0,5), \\ \mathbf{P}(0,1) &= (0,1,3), \\ \mathbf{P}(1,0) &= (1,0,2) \text{ and} \\ \mathbf{P}(1,1) &= (1,1,7) \end{aligned}$$

We can observe that all the iso-parametric lines (x = const. or y = const) are straight.

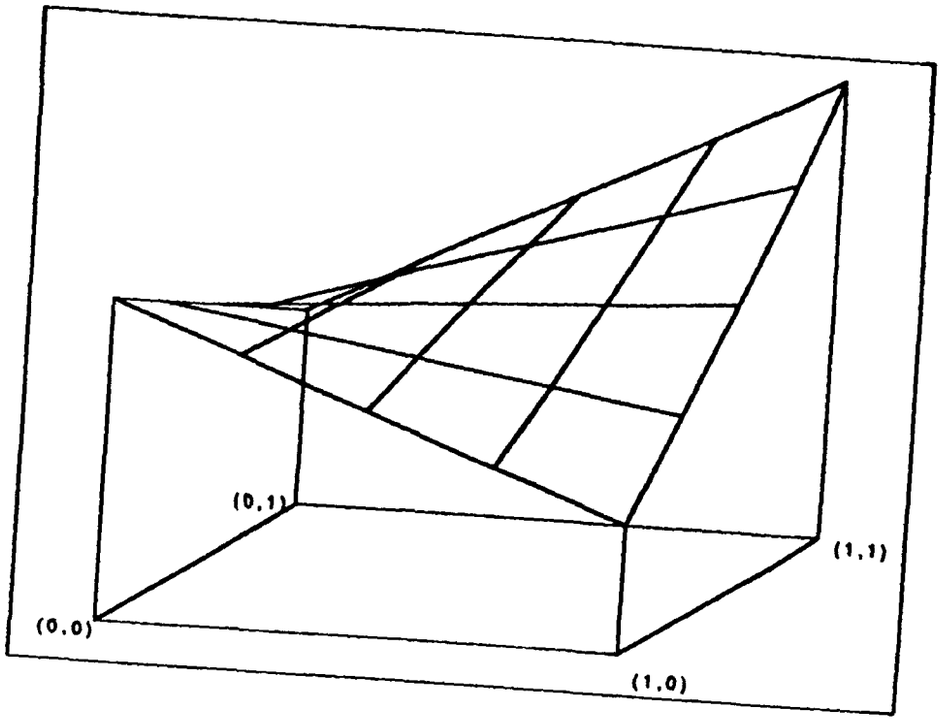


Figure 3.5: A bilinear surface patch

The contour line can be defined by $Q_z(x,y) = P_C$ where P_C is the iso-value. If P_C is inside the range of P_z's, there is a contour line, and the equation for the contour line is

$$\begin{aligned} P_C &= P_z(0,0) - x[P_z(0,0) - P_z(1,0)] - y[P_z(0,0) - P_z(0,1)] \\ &\quad + xy[P_z(0,0) - P_z(0,1) - P_z(1,0) + P_z(1,1)] \end{aligned}$$

We rewrite above equation with substitutions of followings

$$\begin{aligned} P_z(0,0) &= P_{0,0}, \\ P_z(0,1) &= P_{0,1}, \\ P_z(1,0) &= P_{1,0} \text{ and} \\ P_z(1,1) &= P_{1,1} \end{aligned}$$

then

$$y[(P_{0,0} - P_{0,1}) - x(P_{0,0} - P_{0,1} - P_{1,0} + P_{1,1})] = P_{0,0} - x(P_{0,0} - P_{1,0}) - P_C$$

We rearrange it

$$y = \frac{(P_{0,0} - P_C) + x(-P_{0,0} + P_{1,0})}{(P_{0,0} - P_{0,1}) + x(-P_{0,0} + P_{0,1} + P_{1,0} - P_{1,1})}$$

In short form

$$y = \frac{a + bx}{c + dx}$$

where

$$\begin{aligned} a &= P_{0,0} - P_C \\ b &= -P_{0,0} + P_{1,0} \\ c &= P_{0,0} - P_{0,1} \\ d &= -P_{0,0} + P_{0,1} + P_{1,0} - P_{1,1} \end{aligned}$$

and

$$c + dx \neq 0$$

If

$$c + dx = 0,$$

then

$$y = \infty \text{ at } x = -\frac{a}{b}$$

As an example, let's investigate a bilinear patch shown on Figure 3.5 where $P_{0,0} = 5$, $P_{0,1} = 3$, $P_{1,0} = 2$, and $P_{1,1} = 7$. We want to find out contour line(s) of property value of 4. Then,

$$\begin{aligned} a &= 1 \\ b &= -3 \\ c &= 2 \\ d &= -7 \end{aligned}$$

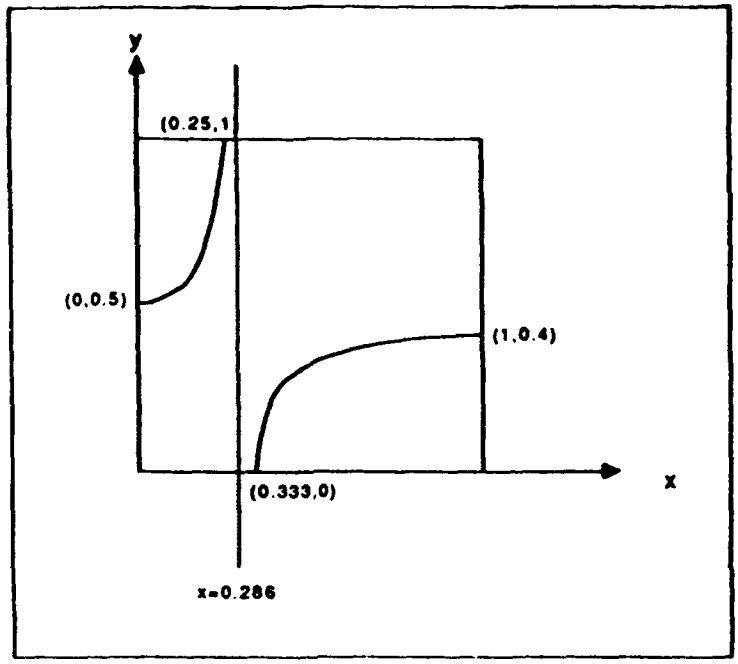


Figure 3.6: Contour lines of a Bilinear surface patch

The contour line is defined by

$$y = \frac{1 - 3x}{2 - 7x}$$

The contour line is shown as Figure 3.6. We can observe that the contour line is not straight whereas the iso-parametric lines are all straight.

Another example case which shows the process of polygonal approximation is generated which is defined by a function

$$P = 0.04xy + z$$

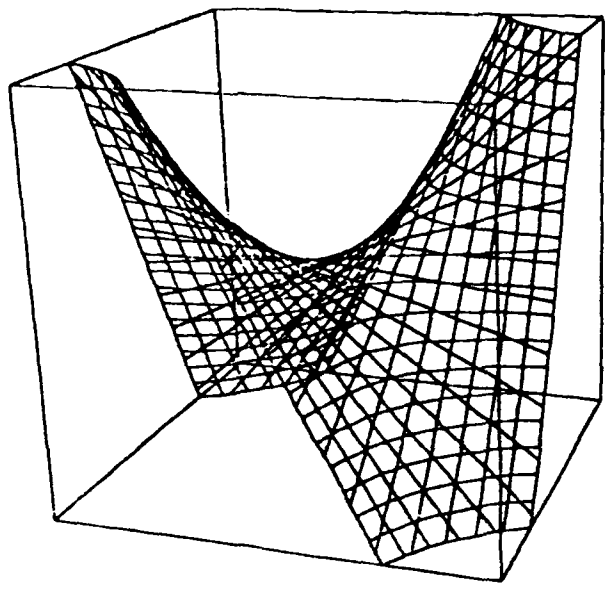


Figure 3.7: Real iso-surface patch

within a box defined by following eight vertices

$$(-50,-50,-50) \quad \text{where} \quad P = +50$$

$$(+50,-50,-50) \quad \text{where} \quad P = -150$$

$$(-50,+50,-50) \quad \text{where} \quad P = -150$$

$$(+50,+50,-50) \quad \text{where} \quad P = +50$$

$$(-50,-50,+50) \quad \text{where} \quad P = +150$$

$$(+50,-50,+50) \quad \text{where} \quad P = -50$$

$$(-50,+50,+50) \quad \text{where} \quad P = -50$$

$$(+50,+50,+50) \quad \text{where} \quad P = +150$$

Figure 3.7 shows the box and an iso-surface of value $P_C = 0$ within the box. The iso-surface will be approximated to a sum of triangles by the algorithm developed in this report. First, the intersection points of the iso-surface with edges of the box will be approximated by linear interpolation along each edge. On the edge E_1 (see

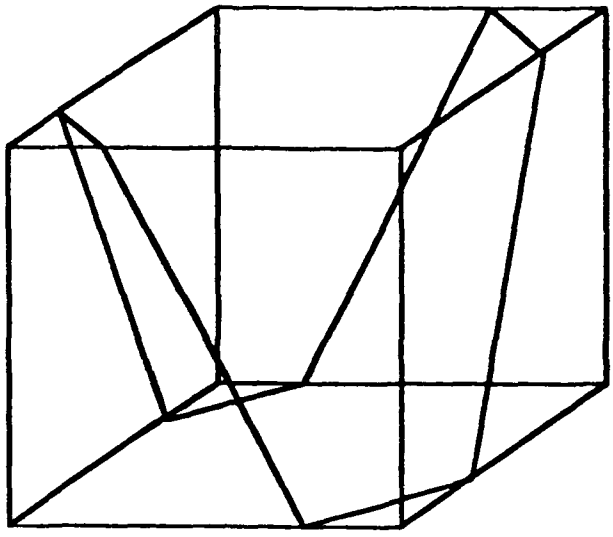


Figure 3.8: Approximated iso-surface patch

Figure 3.2), $y = -50$ and $z = -50$. The exact x value of the intersection point is calculated by

$$0 = 0.04x(-50) + (-50)$$

and

$$x = -25$$

The linear interpolation gives x value of the intersection point as

$$x = -25$$

The top face and bottom face have four intersection points each. By connecting these intersecting points with straight lines, we approximate the iso-surface patch within the box as figure 3.8. This non-planar 8-sided polygon will be divided into triangles later.

Chapter 4

CLASSIFICATION OF BOX-SURFACE INTERSECTIONS

As can be seen from Figures 3.3 and 3.8, an iso-surface may intersect a selected box in various ways. Before an algorithm can be devised to find the boundaries of the surface patches (in the form of non-planar polygons), the possible topologies of the polygons must be investigated individually.

4.1 The Total Number of Intersection Cases

The points S of the polygons are found by intersecting the iso-surface with the edges of the box. Assuming a linear change of the property value along each edge, the intersection point is found from

$$\mathbf{S} = \mathbf{V}_1 + (\mathbf{V}_2 - \mathbf{V}_1) \times \frac{P_2 - P_C}{P_2 - P_1} \tag{4.1}$$

where, $\mathbf{V}_1$ and $\mathbf{V}_2$ denote the vertices at either end of a selected edge, P_1 and P_2 are the corresponding properties given at $\mathbf{V}_1$ and $\mathbf{V}_2$, respectively. Figure 4.1 illustrates the finding of S along the edge.

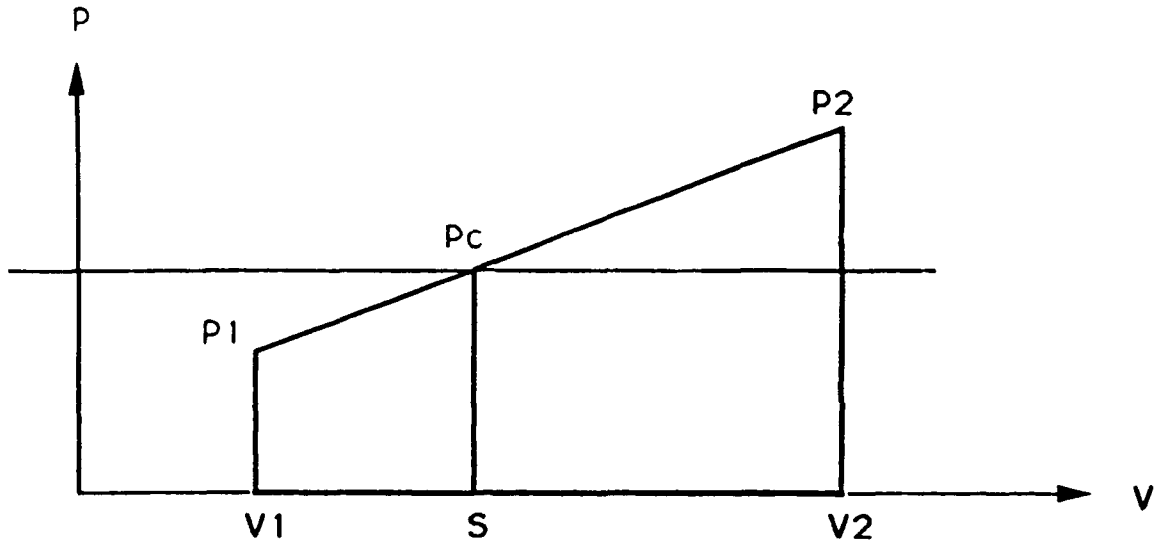


Figure 4.1: Linear interpolation along an edge

The different patterns of box-surface intersection are determined by the number of *edge intersections* found and by the topological configuration of the intersection points. An edge may have no intersection point at all if

$$P_C > P_1 \text{ and } P_C > P_2$$

or

$$P_C < P_1 \text{ and } P_C < P_2$$

An edge may have one intersection point somewhere in the middle, may have a single intersection point at either of both limiting vertices, or the entire edge may lie completely on the iso-surface if $P_C = P_1 = P_2$.

The various edge intersections are created by different combinations of property values at the eight vertices. As obvious from the above, only three conditions at each vertex are significant: the property at a vertex is either grater than P_C, is less than

P_C, or is identical with P_C. Therefore, a total of

$$3^8 = 6,561$$

cases are possible for a single box. Because of symmetry, many of these 6,561 cases can be handled as identical patterns. Nevertheless, the number of remaining cases is too large for the development of an effective algorithm.

4.2 Reduced Number of Intersection Cases

The possible number of box-surface intersection can be significantly reduced by eliminating the condition that a property at a vertex is identical with P_C. In practical applications, the probability for an equality condition is extremely small. In the rare case that such an equality occurs, a vertex property which is identical with P_C can be modified (for example, lowered) by a very small amount. The algorithm uses the following method for a consistent property modification:

```
      PTOL = (Pmax - Pmin) * 0.00001
c          10 * min. positive floating point value with 4 bytes
      r1min = 1.E-37
      IF (PTOL .LT. r1min) PTOL= r1min
c
      IF ( ABS(P1-PC) .LT. PTOL) P1 = PC - 0.5*PTOL
```

Figure 4.2 shows a result of such modification for a vertex with four neighboring faces.

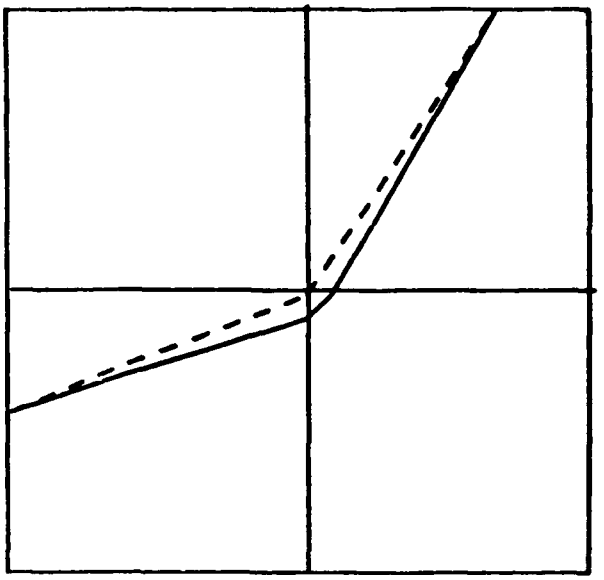


Figure 4.2: Slight modification of property value

After this modification, only two significant conditions remain for a vertex: the property value is either greater than P_C or is less than P_C. Therefore, a total of

$$2^8 = 256$$

intersection cases remain and need to be investigated. It also can be considered as *bit patterns* of 8 bits. Each bit represents each vertex of a box and it can have two values, 0 and 1, depending on the property values there.

4.3 Classification in Groups

A study of all the 256 cases of box-surface intersections shows that the number of edge intersections can be used to classify the cases into nine different groups. Table 4.1 shows those 9 groups with their frequencies and unique patterns. The minimum number of edge intersections is zero, and the maximum is twelve. Note

No. of intersections	No. of cases toward 256	No. of unique patterns
0	2 cases	1
1	-	-
2	-	-
3	16 cases	1
4	30 cases	1
5	48 cases	1
6	64 cases	2
7	48 cases	2
8	30 cases	3
9	16 cases	3
10	-	-
11	-	-
12	2 cases	4
Total	256 cases	18

Table 4.1: Grouping of Cases of box-surface intersections

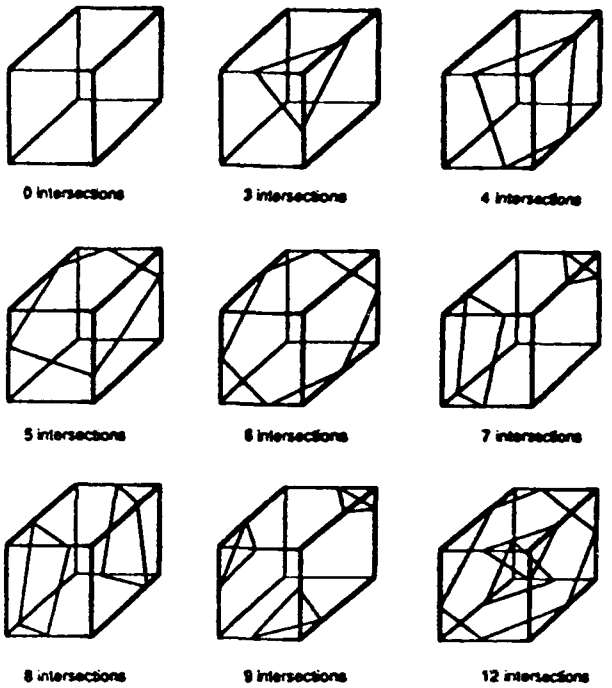


Figure 4.3: Sample cases of Nine groups

that cases of 1, 2, 10, and 11 edge intersections do not exist. Figure 4.3 shows sample cases of nine groups. All the 256 cases grouped by the number of edge intersections are enumerated in Appendix A.

In the following, possible connecting patterns for each group are identified. One *pattern* is identified as unique by its topology, i.e. number of its sides (edges) and number of separate parts within a box. Figure 4.4 shows those unique patterns of each group.

- Group 1 : No. of edge intersections = 0 ; 2 cases of the 256 cases have no edge intersections. All values of 8 vertices are higher than the iso-surface value or lower than that. One case is the *dual* case of the other. The bit pattern of 00000000 produces the same topological pattern with the bit pattern of 11111111. In these cases, no surface patches or polygons exists, but it is counted as one pattern.

(see the following page 28.1)

Figure 4.4: Unique patterns of iso-surface patches of 9 groups

- Group 2 : No. of edge intersections = 3 ; 16 cases of the 256 cases have 3 edge intersections. In these cases, there is only one unique pattern. The pattern is formed by connecting 3 edge intersections in any order. They form a triangle.

- Group 3 : No. of edge intersections = 4 ; 30 cases have 4 edge intersections. There is one unique pattern. Four edge intersections form a 4-sided polygon.

- Group 4 : No. of edge intersections = 5 ; 48 cases have 5 edge intersections. There is only one pattern. Five edge intersections form one 5-edged polygon.

- Group 5 : No. of edge intersections = 6 ; 64 cases have 6 edge intersections. There are two different patterns. One of them is 6-edged single polygon. The other pattern is made of two triangles.

- Group 6 : No. of edge intersections = 7 ; 48 cases have 7 edge intersections.

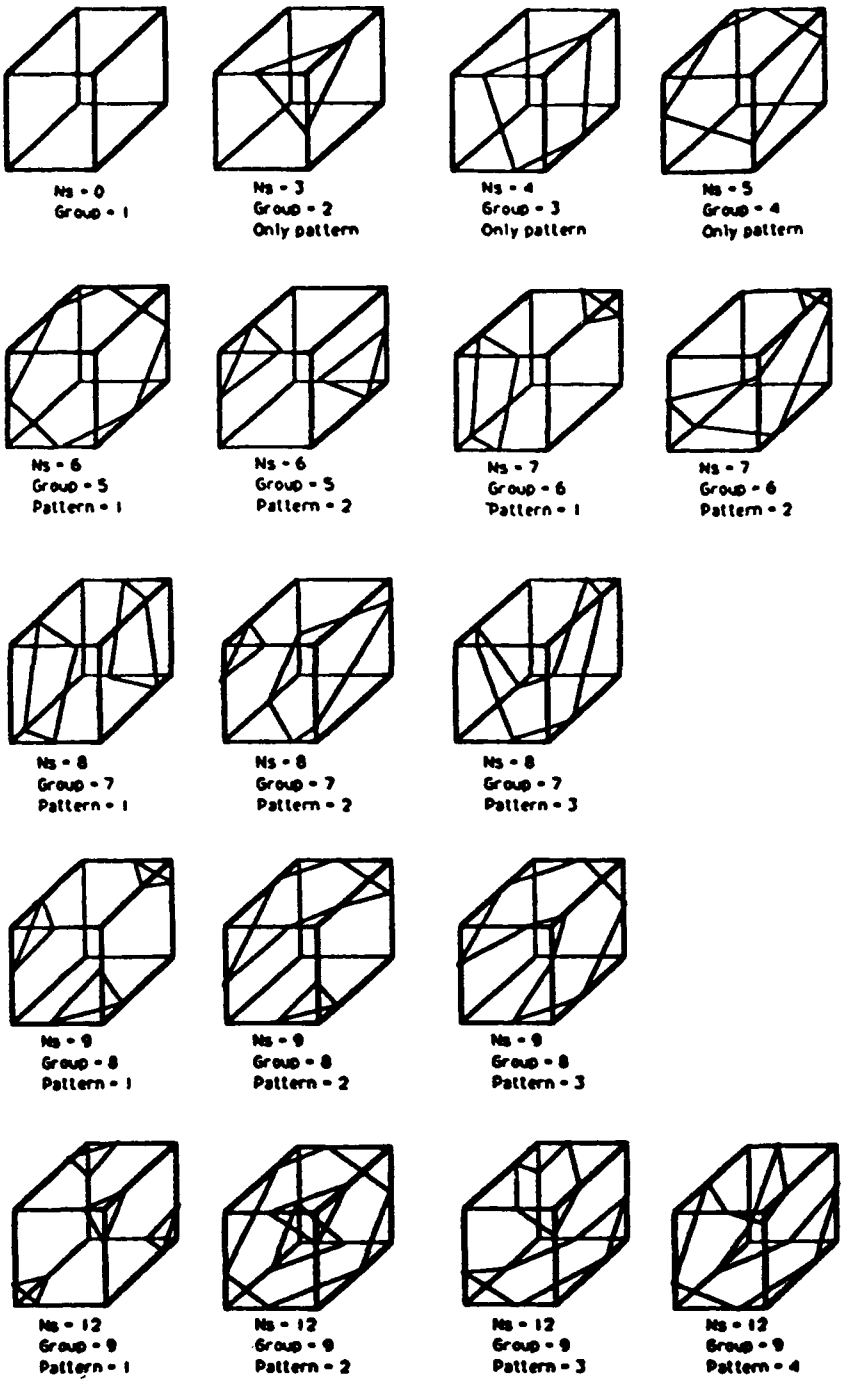


Figure 4.4: Unique patters of iso-surface patches of 9 groups

There are two patterns. One of them is 7-edged single polygon. The other has one triangle and one quadrilateral.

- Group 7 : No. of edge intersections = 8 ; 30 cases have 8 edge intersections. There are three different patterns. One of them are 8-edged single polygon. Another pattern has one triangle and one 5-edged polygon. Still another has two quadrilaterals.

- Group 8 : No. of edge intersections = 9 ; 16 cases has 9 edge intersections. There are three patterns. One of them is composed of 9-edged single polygon. Another has one triangle and a 6-edged polygon. The last pattern is composed of three triangles.

- Group 9 : No. of edge intersections = 12 ; 2 cases have 12 edge intersections. There are four patterns. 1) Four triangles. 2) Two triangles and one 6-edged polygon. 3) Two 6-edged polygons. 4) One 12-edged polygon.

Chapter 5

HANDLING OF AMBIGUOUS CASES

If any of 6 faces of a grid box has 4 edge intersections, there is no unique way of connecting edge intersections on the face. This produces ambiguities on the algorithm. One must find the correct way of connecting intersection points. For a face, there are 3 possible ways of connection. Figure 5.1 shows a face with ambiguity and possible connections.

One simple solution is to subdivide the domain (i.e. a box) into smaller cells until the ambiguity is dissolved. This is called as *adaptive subdivision*, or *tesselation*. But this method requires functional values at internal points of one grid box, which are not directly availabe in a given discrete data set. That requires interpolation of proper form (local or global). Also, it creates cracks on an iso-surface between different size cells, i.e., between subdivided box and undivided box. Because straight line is used to connect the intersection points, the side of the iso-surface patch found on the smaller size box does not match with that on the bigger size box. The subdivided box has more intersection points and they approximate the iso-surface more closely. Figure 5.2 shows one face shared by two boxes, one of them is subdivided into eight

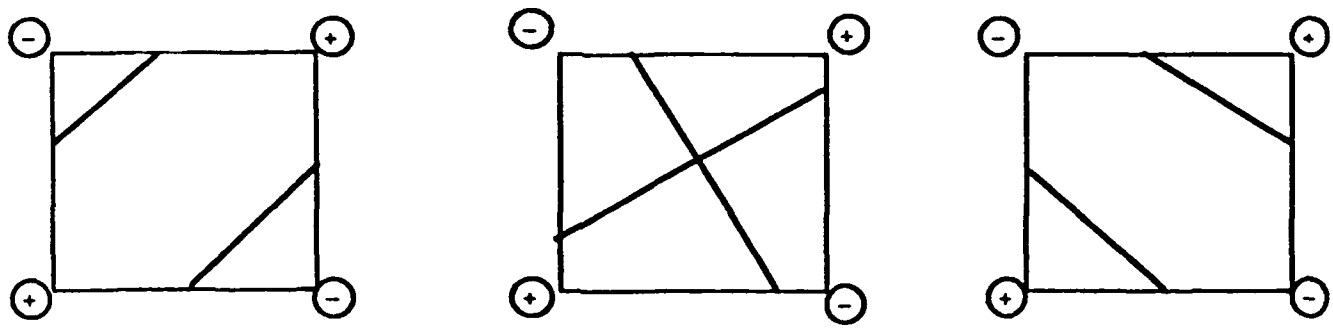


Figure 5.1: A face with Ambiguity

smaller boxes.

Triangulation of the grid box is another solution. A triangular face can have exactly two intersections, if any. There is only one way of connecting them. Figure 5.3 shows a triangular face. In general, no ambiguities exists if a simplex is the partitioning cell. A *simplex* is the simplest linear decomposition of n-space. For 3-D space, it is tetrahedron. Figure 5.4 shows possible patterns within a tetrahedron. This method requires triangulation of domain (into tetrahedra), and also requires function values at number of internal points. Finding intersection points along edges of triangulated cell is compuationally more expensive than along edges of box shaped cells, where edges can be aligned with coordinate axes.

Still another solution is utilizing function values at the center of ambiguous faces. From the fact that the iso-surface divides the higher value region from the lower value

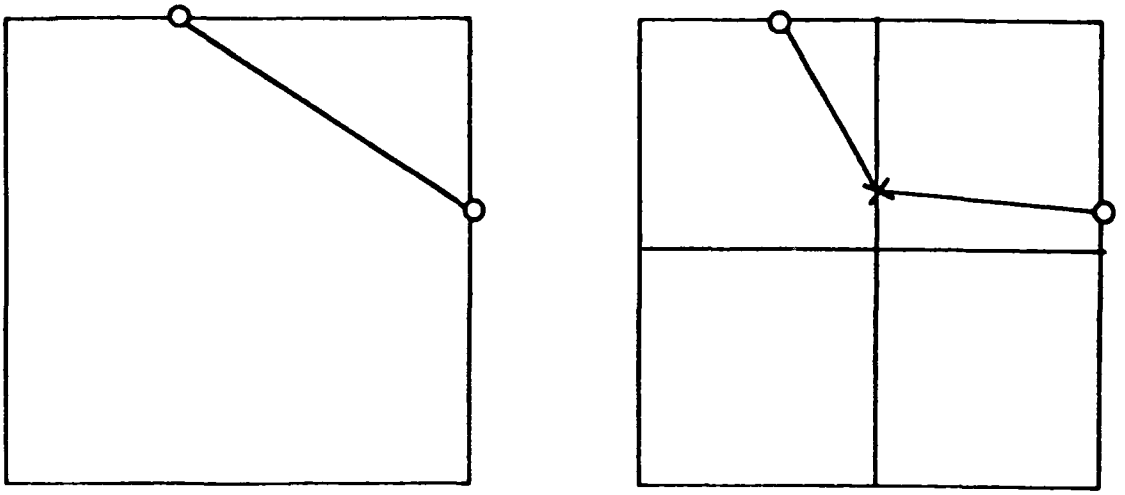


Figure 5.2: Cracks between different size cells

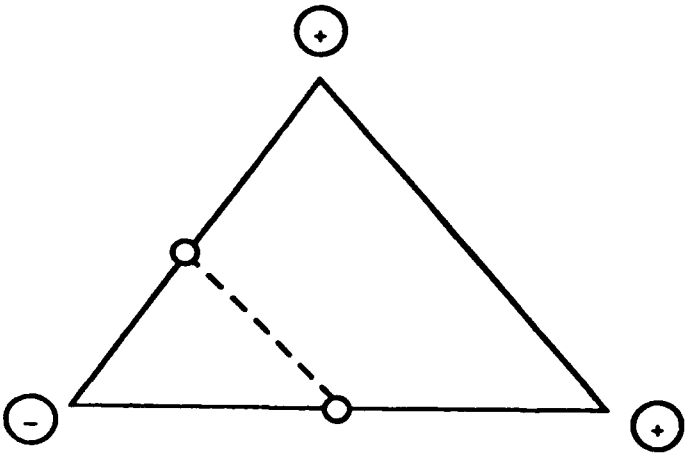


Figure 5.3: A triangular face

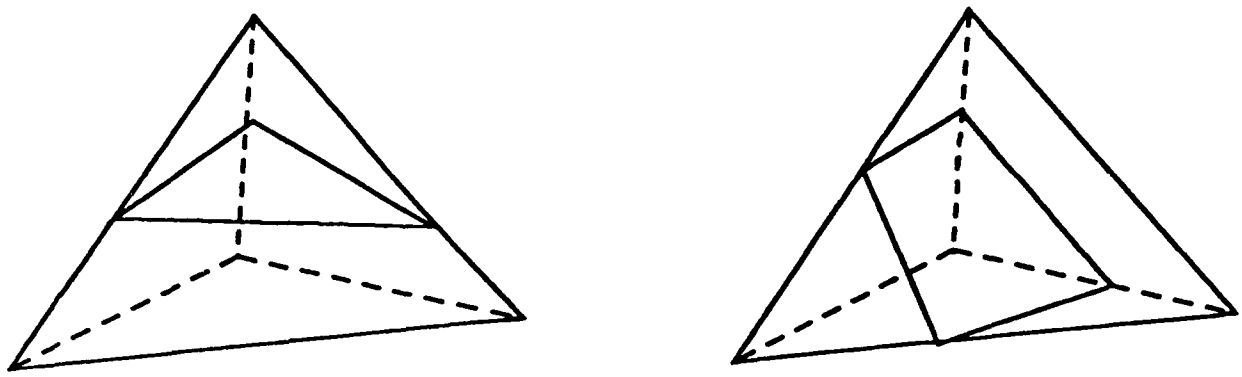


Figure 5.4: Iso-surface patterns within Tetrahedra

region, the correct way of connecting points can be determined. The function value at the center of ambiguous face is tested against the iso-value P_C. If the centroidal value is higher than P_C, the two vertices with lower values will be isolated, vice versa. Figure 5.5 shows a face with four edge intersections and the centroidal value. This method gives the same result as when the box is subdivided into 24 tetrahedra, if additional information is only used to get the correct connecting information, i.e., additional intersections created inside and on the box are neglected. This method still needs function value at the centroid of the ambiguous face. Some form of interpolation should be applied.

Linear interpolation (specifically, mean of values of 4 vertices) is used in the implementation to get the center value. It is a simplified form of the *distance weighted* interpolation. The example case shown in Figure 3.8 is constructed by using the

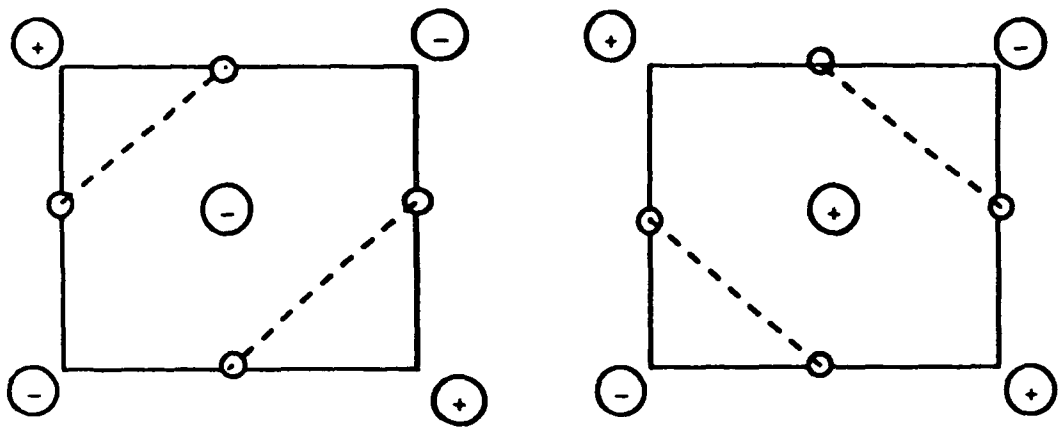


Figure 5.5: Four edge intersections and the Cetroid value

mean value to get the connecting information on top and bottom faces. If the mean of property values of four vertices is used, the centroid value of the top face is higher than the iso-value in Figure 3.8. That of the bottom face is lower than the iso-value.

As linear interpolation is also used for finding edge intersections, it is consistent throughout the domain. It will produce consitent results at neighboring grid boxes too, which share the same ambiguous face. The resulting iso-surface will be an unique surface.

Chapter 6

ALGORITHM FOR CREATION OF ISO-SURFACE PATCHES

An algorithm for connecting the edge intersections to construct the polygonal iso-surface patches has been developed.

6.1 The Logic of the Algorithm

The algorithm treats every grid box identically. It starts from investigating each of 6 faces of a box. For each face, the algorithm finds edge intersections, if any, by comparing property values at the vertices against the iso-value P_C. Then, it establishes the connecting information among the points found on each face. If two points are found, the connection is trivial. If four points are found, it calculates the centroidal value to find the correct way of connecting points. Those intersection points are saved to an array with their coordinates, face identification, edge identification, and connecting information. The local identification system shown on Figure 3.2 is used. Table 6.1 shows an example of the data structure of the array.

After investigating all six faces of the box, the algorithm sorts the saved edge intersections to construct polygons. We have the number of edge intersections with

	Edge Id	Face Id	Connecting edge	Coordinates		
				x	y	z
S_1	E_7	F_6	E_6	.	.	.
S_2	E_6	F_6	E_7	.	.	.
S_3	E_6	F_3	E_{11}	.	.	.
S_4	E_{11}	F_3	E_6	.	.	.
S_5	E_{11}	F_4	E_7	.	.	.
S_6	E_7	F_4	E_{11}	.	.	.

Table 6.1: Data structure for the algorithm

their coordinates, face id's, edge id's, and connecting information on face level. But, each edge intersection is saved twice from the previous process because one edge is shared by two faces. Starting from any point, it moves to the connecting point on the same face. Then it connects the duplicated point on the same edge. It travels along the sides of an iso-surface patch until it returns to the strating point. If there are points remained, not connected, it starts from one of the remainders and does the same procedure to find another patch within the grid box. Finally the iso-surface patches are triangulated for the rendering process. Figure 6.1 shows the sorting process, which travels along the boundary of an iso-surface patch. The previous Table 6.1 shows the data structure after the sorting process. The algorithm in pseudo-code is as follows.

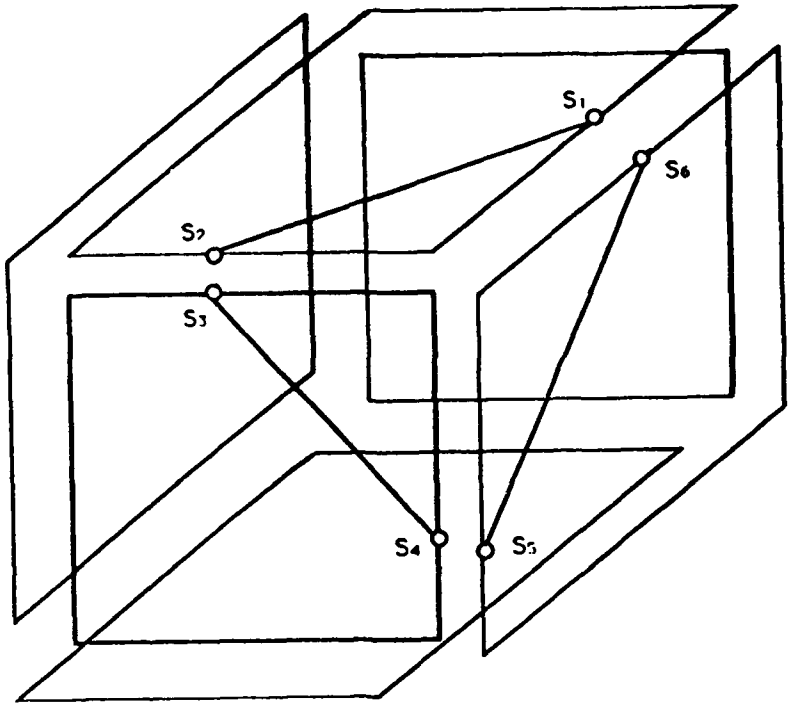


Figure 6.1: Traveling along a polygon

For all faces of one box

Find all edge intersections, if any

Find connecting information on the face

If 2 points : trivial

If 4 points : use function value of centroid to get connecting info.

Save their coordinates with face id, edge id,

and face level connecting information

Sort edge intersections to construct polygon(s)

Triangulate each polygon found

End

Procedure : Sort edge intersections to construct polygon(s)

Start from any point

Find connecting point on the same face

Find duplicated point on the same edge

Keep going until it returns to the starting point

If there are points remained, not connected, start one of them and

do the same work as above

End

Observations used throughout the algorithm are summarized as follows .

- A box has 6 faces, 8 vertices, and 12 edges.
- One edge can have at most one intersection with an iso-surface. It is because of the linear interpolation used along each edge of the grid box.
- One face can have 0, 2, or 4 edge intersections.
- One box can have at most 12 intersections, one per 12 edges of the box.
- Within one grid box, there may be more than one patch of the iso-surface.
- Iso-surface divides the higher value region from the lower value region. The surface has inside and outside information.
- Polygonal iso-surface patches are non-planar in most cases.

6.2 Calculation of Surface Normals

Normal vectors of surface patches are required for the rendering process such as Gouraud shading or Phong shading. They can be calculated by taking cross product

of edge vectors of polygonal patches which share a vertex. This method of calculating normal vectors gives rough approximation. Average of normal vectors calculated from all polygons sharing a vertex is usually used for the final rendering process.

Alternatively, numerical derivatives can be used, based upon the fact that the directional derivative at a point is the normal vector of iso-surface at the point. For $F(x, y, z) = 0$ is given, ∇F is the normal vector at the point. The normal vector (directional derivative) points toward the higher value region, so inside and outside information of the iso-surface is available.

Gradients are estimated by central difference method. The cases where intervals are not same has been considered. For x coordinate direction, the forward difference is

$$f'_f(x_i) = \frac{f(x_{i+1}) - f(x_i)}{\Delta x_f}$$

The backward difference is

$$f'_b(x_i) = \frac{f(x_i) - f(x_{i-1})}{\Delta x_b}$$

The central difference is

$$f'_c(x_i) = \frac{\Delta x_f \cdot f'_b(x_i) + \Delta x_b \cdot f'_f(x_i)}{\Delta x_b + \Delta x_f}$$

Figure 6.2 shows the process of taking the numerical gradient. The other y, z components of ∇F can be obtained by the same procedure.

At boundaries of the computational domain, the central differences cannot be calculated. Forward differences or backward differences are used as the normal vectors instead. These normal vectors are then normalized against its own length to get the *unit* normal vectors.

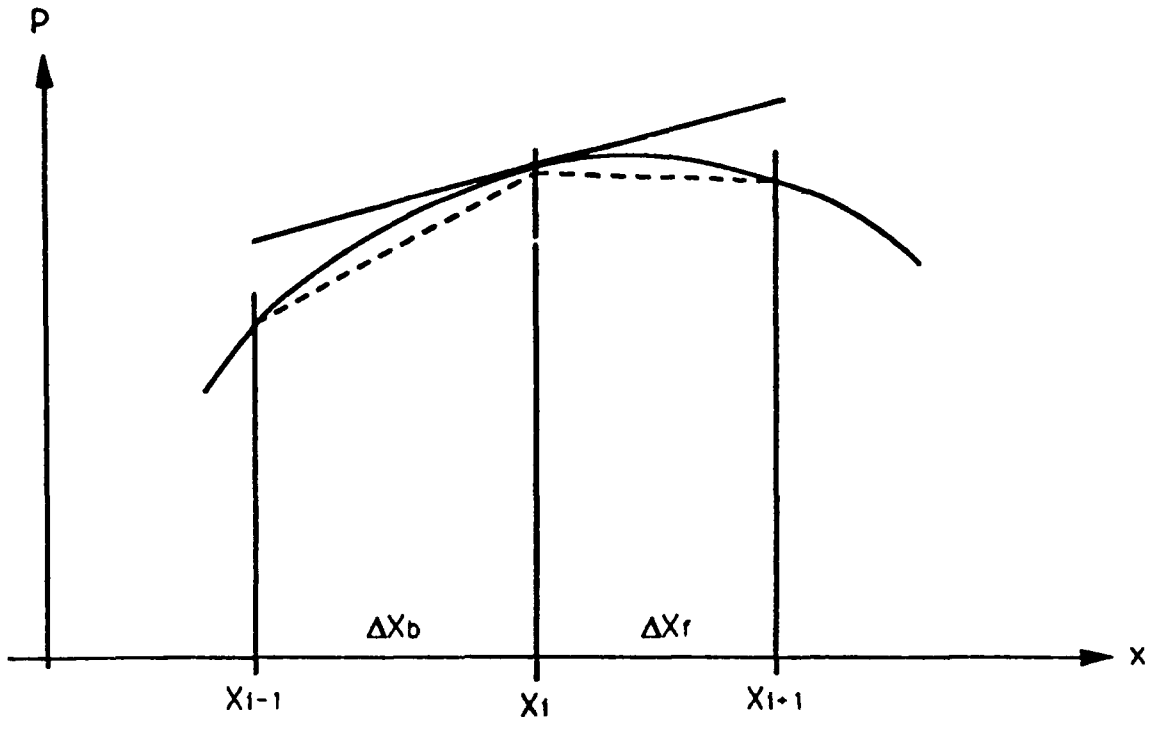


Figure 6.2: Numerical differentiations

6.3 Implementation of the Algorithm

Hierarchical data structure is used for handling of the adjacency information of one box. See Appendix B. Topological queries which ask neighboring faces, edges, and vertices can be answered through this data structure.

Implementation of the algorithm is done on a Stellar GS1000 series graphics super computer. Phigs+ implementation on the Stellar Computer, which is based on X-window system, is used throughout as the main graphics package. Fortran77 is the main programming language. Stellar's implementation of Phigs+ is not complete, and its baseline graphics are X-window system written in language C, and a few C procedures are used too. Inter-language calling conventions for C and Fortran, which is set by Stellar Computer, are used.

After the polyhedral approximation of the iso-surface is constructed, rendering of

the surface is handled by the general purpose visualization system - AVS (Application Visualization System) - supplied by Stellar Computer.

Chapter 7

APPLICATION AND CONCLUSION

The developed algorithm can be used for a range of areas as long as the required data set can be supplied. It can be used to visualize fluid flows from the result of Computational Fluid Mechanics, Meteorological data such as clouds and storm, Seismic data for mineral resource development, Medical imaging for human organs, ... etc.

As the iso-surface has inside - outside information inheritantly, the developed algorithm can be extended to reconstruct solid object. That is, topological neighboring information required by a solid modeling system can be extracted through application of the method to a regularly sampled data set. One good example is reconstruction of a human organ from a CT (Computed Tomograph) scan data set.

Some assumptions are made in the development of the algorithm based on observations. They should be tested or proved rigorously. Those assumptions are as follows.

Iso-surface divide hot region (higher value region) from cool region (lower value region) completely. There should not be any holes on the iso-surface between above

two regions, if continuity of function in the domain is assumed. At one instant of time, the iso-surface divide the two regions like as a solid surface.

Can the algorithm completly covers the Iso-surface ? As the method use only local information available on one grid box, and does not consider any neighboring information or globality, its ability to reconstruct the whole iso-surface completely can be suspected. Only observation which enforce the global connectivity is that two neighboring boxes use the same information (intersection points and connection of them). The same information is used by the neighboring box for the shared face. See Figure 3.4

The developed algorithm is relatively simple in that only local information is used for the construction of iso-surface. This characteristic can be used for speeding up via parallel processing. The algorithm is also found to be robust.

Bibliography

[1] R. E. Barnhill, G. T. Makatura, and S. E. Stead. A new look at higher dimensional surfaces through computer graphics. In Farin [9].

[2] K. P. Beier. *Viewx - Isox : Preliminary Documentation (version 3.2).* Univ. of Michigan, Dept. of Naval Architecture and Marine Engineering, 1985.

[3] K. P. Beier. *Computer-Aided Hull Design and Production.* Dept. of Naval Architecture and Marine Engineering, Univ. of Michigan, Fall 1988. Informal Lecture Notes (NA574).

[4] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transaction on Graphics*, 1(3):235–256, 1982.

[5] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, (5):341–355, 1988.

[6] J. D. Boissonnat. Reconstruction of solids. In *Symposium on Computational Geometry*, pages 46–54, 1985.

[7] H. E. Cline, W. E. Lorensen, S. Ludke, C. R. Crawford, and B. C. Teeter. Two algorithms for the three-dimensional reconstruction of tomograms. *Medical Physics*, 15(3):320–327, 1988.

[8] R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *Computer Graphics*, 22(4):65–74, 1988. ACM Siggraph.

[9] G. E. Farin, editor. *Geometric Modeling: Algorithms and new trends.* SIAM (Society for Industrial and Applied Mathematics), 1987.

[10] W. L. Hibbard. 4-D display of meteorological data. In F. Crow and S. M. Pizer, editors, *1986 Workshop on Interactive 3D Graphics*, Univ. of North Carolina.

[11] J. S. Hojnicki and P. R. White. Converting CAD wireframe data to surfaced representations. pages 19–25, Mar/Apr 1988.

[12] K. Kaneda, K. Harada, E. Nakame, M. Yasuda, and A. G. Sato. Reconstruction and semi-transparent display method for observing inner structure of an object consisting of multiple surfaces. *The Visual Computer*, (3):137–144, 1987.

[13] W. Kaplan. *Advanced calculus.* Addison Wesley, 3rd edition, 1984.

[14] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987. ACM Siggraph.

[15] C. S. Petersen, B.R. Piper, and A. J. Worsey. Adaptive contouring of a trivariate interpolation. In Farin [9].

[16] T. W. Sederberg. Algebraic geometry for surface and solid modeling. In Farin [9].

[17] C. Upson. The visual simulation of amorphous phenomena. *The Visual Computer*, (2):321–326, 1986.

[18] T. Wright and J. Humbrecht. ISOSRF - an algorithm for plotting iso-valued surfaces of a function of three variables. *Computer Graphics*, 13(2):182–189, 1979. ACM Siggraph.

[19] G. Wyvill, C. McPheeters, and B. Wyvill. Data structure for *soft* objects. *The Visual Computer*, (2):227–234, 1986.

Appendix A

256 cases of Box-Surface Intersections

All the 256 cases of box-surface intersections are enumerated. They are grouped by the number of edge intersections.

Appendix A

The following pages show all the 256 cases of box-surface intersection. The cases are sorted by the number of edge intersections (IN) resulting in 9 different groups.

Explanation:

+	indicates $P > P_c$ at vertex
-	indicates $P < P_c$ at vertex
•	indicates edge intersection point

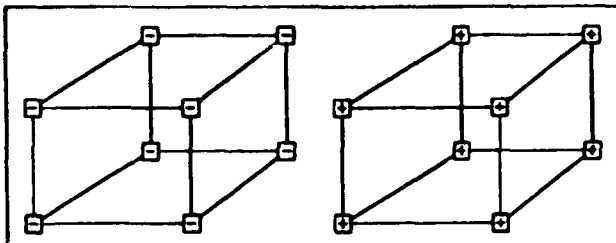
ID = n=bbbbbbbb

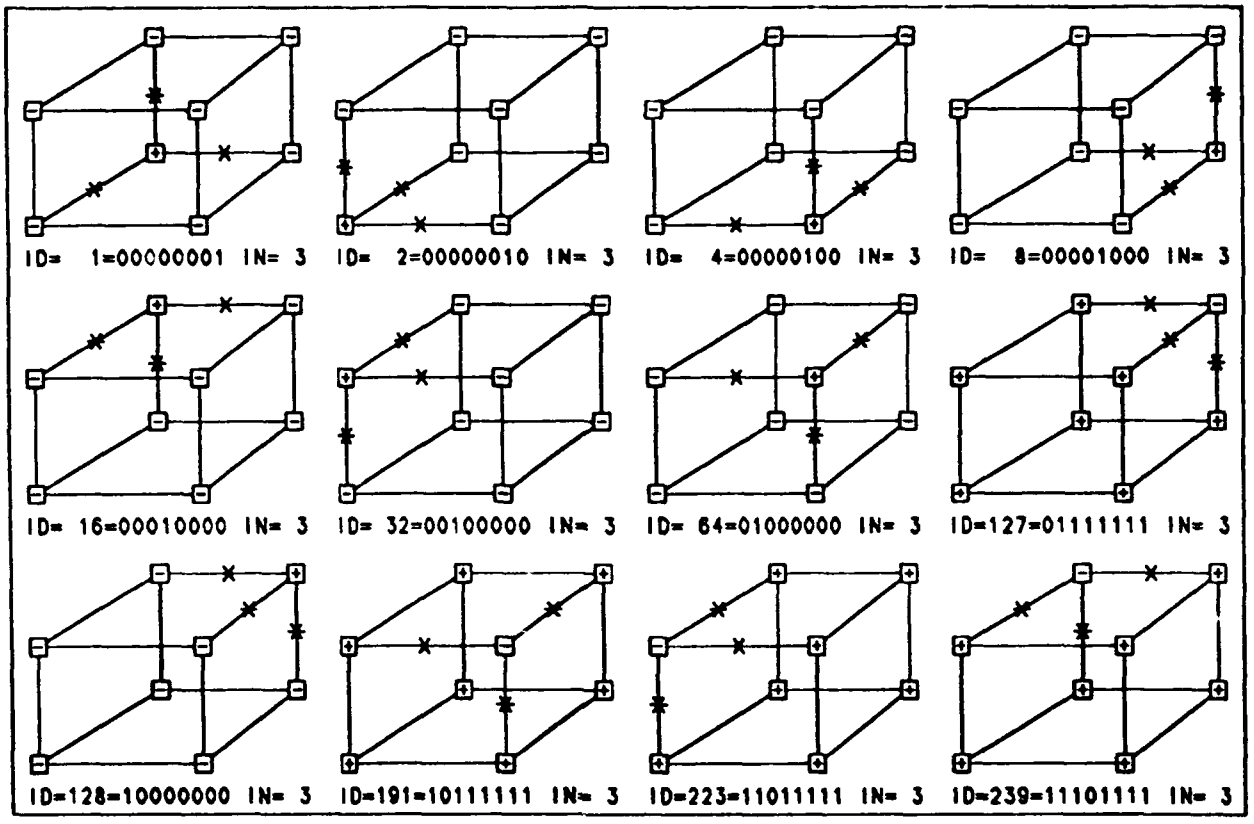
n = case ID number (between 0 and 255)

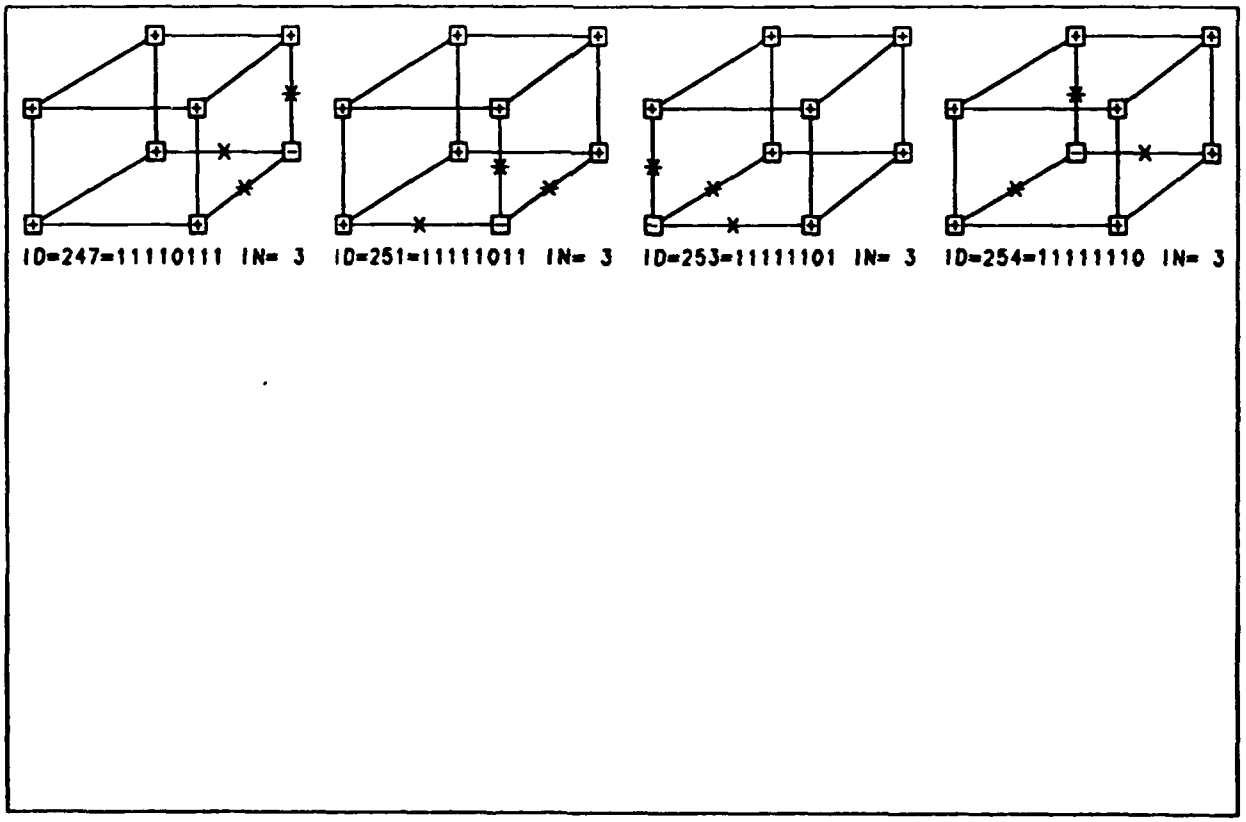
bbbbbbbb = corresponding bit pattern

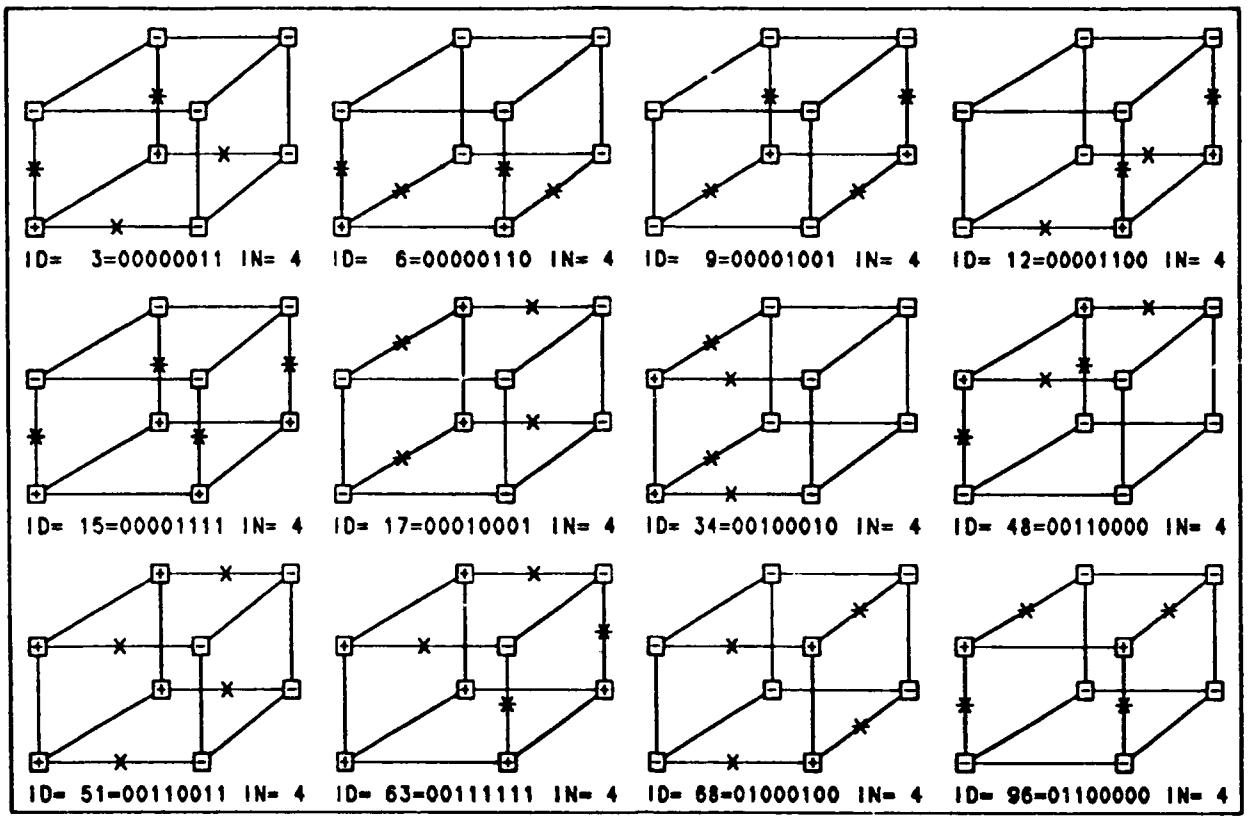
IN number of edge intersections

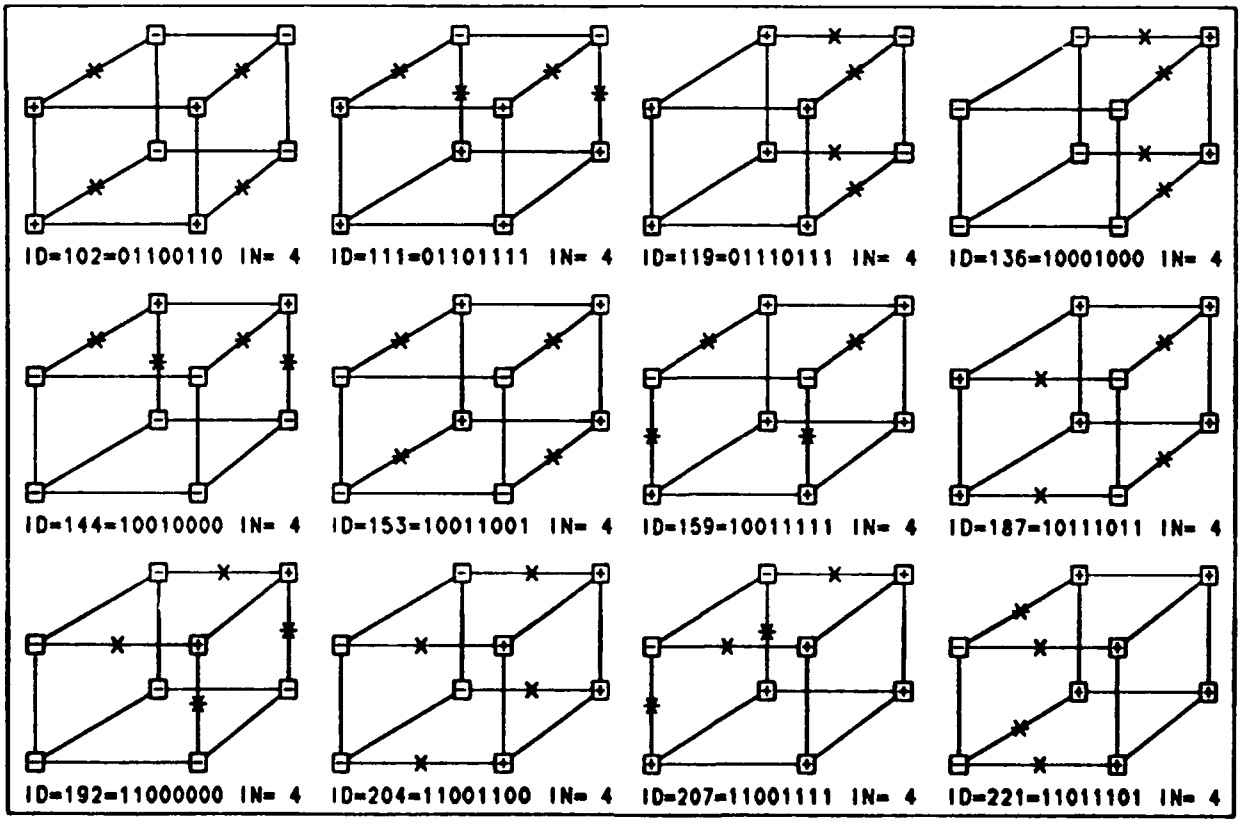
Directory:

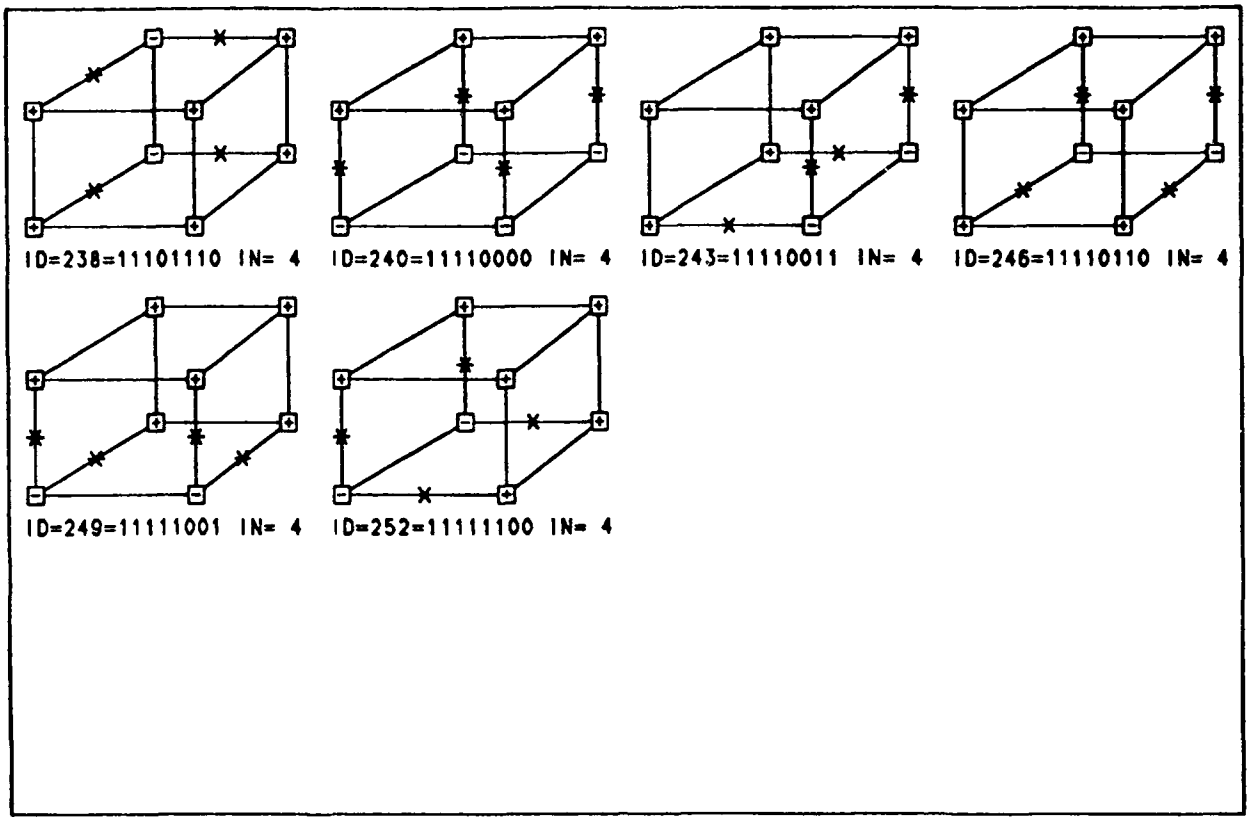

ID= 0=00000000 IN= 0
ID=255=11111111 IN= 0

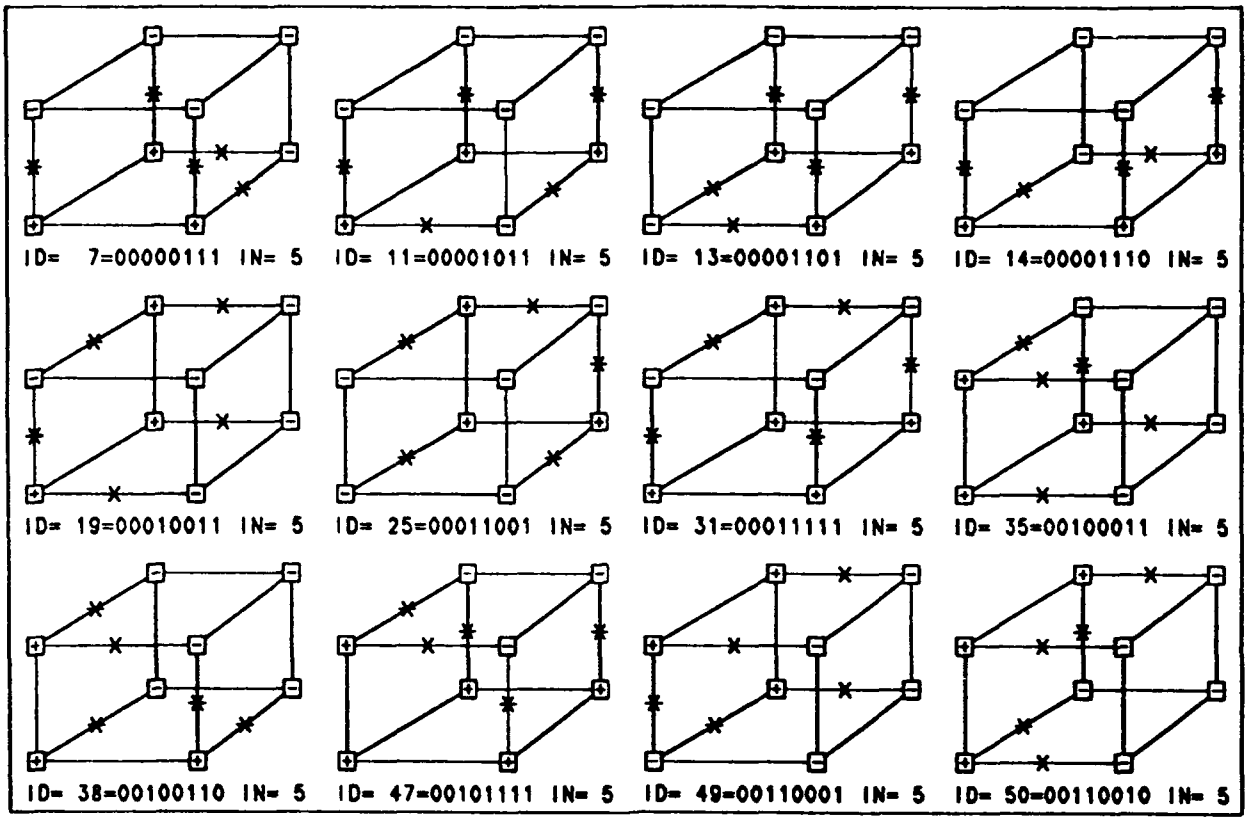

ID= 1=00000001 IN= 3
ID= 2=00000010 IN= 3
ID= 4=00000100 IN= 3
ID= 8=00001000 IN= 3
ID= 16=00010000 IN= 3
ID= 32=00100000 IN= 3
ID= 64=01000000 IN= 3
ID=127=01111111 IN= 3
ID=128=10000000 IN= 3
ID=191=10111111 IN= 3
ID=223=11011111 IN= 3
ID=239=11101111 IN= 3

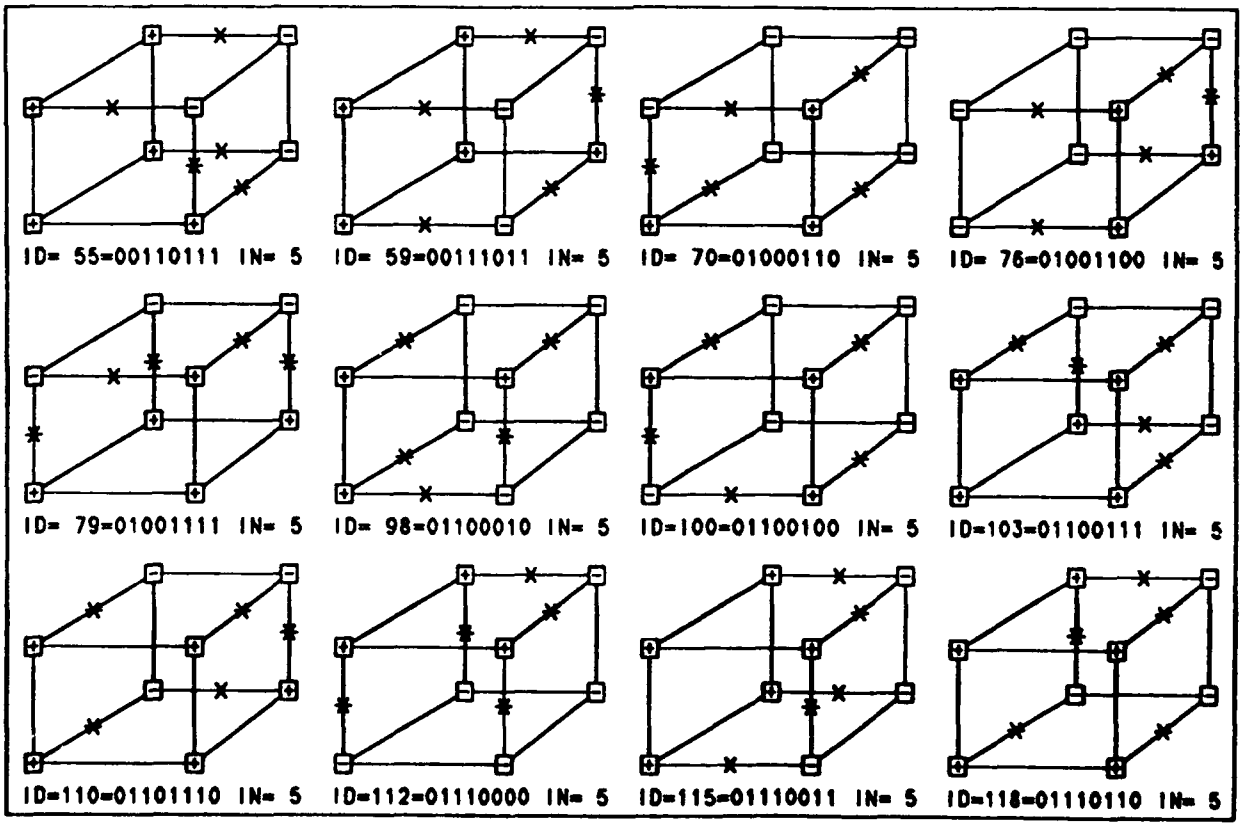

ID=247=11110111 IN= 3
ID=251=11111011 IN= 3
ID=253=11111101 IN= 3
ID=254=11111110 IN= 3

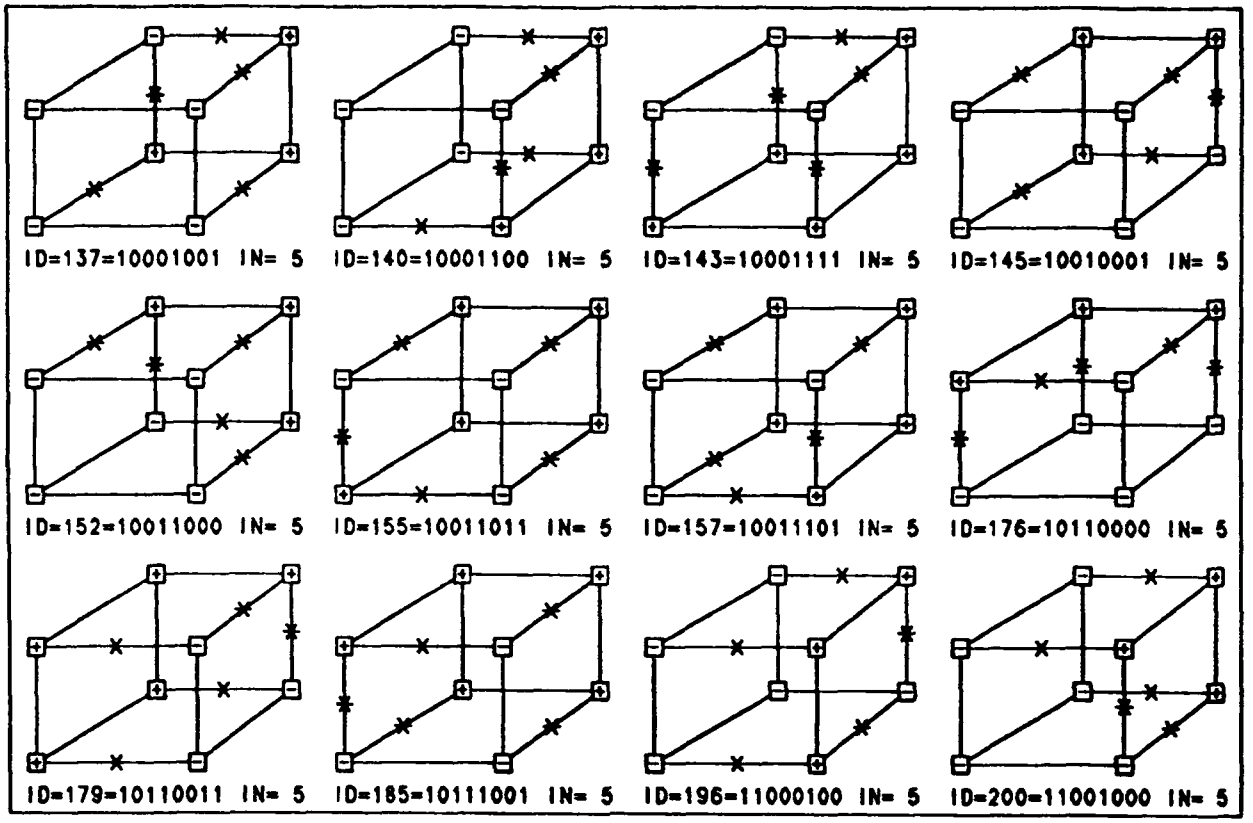

ID= 3=00000011 IN= 4
ID= 6=00000110 IN= 4
ID= 9=00001001 IN= 4
ID= 12=00001100 IN= 4
ID= 15=00001111 IN= 4
ID= 17=00010001 IN= 4
ID= 34=00100010 IN= 4
ID= 48=00110000 IN= 4
ID= 51=00110011 IN= 4
ID= 63=00111111 IN= 4
ID= 68=01000100 IN= 4
ID= 96=01100000 IN= 4

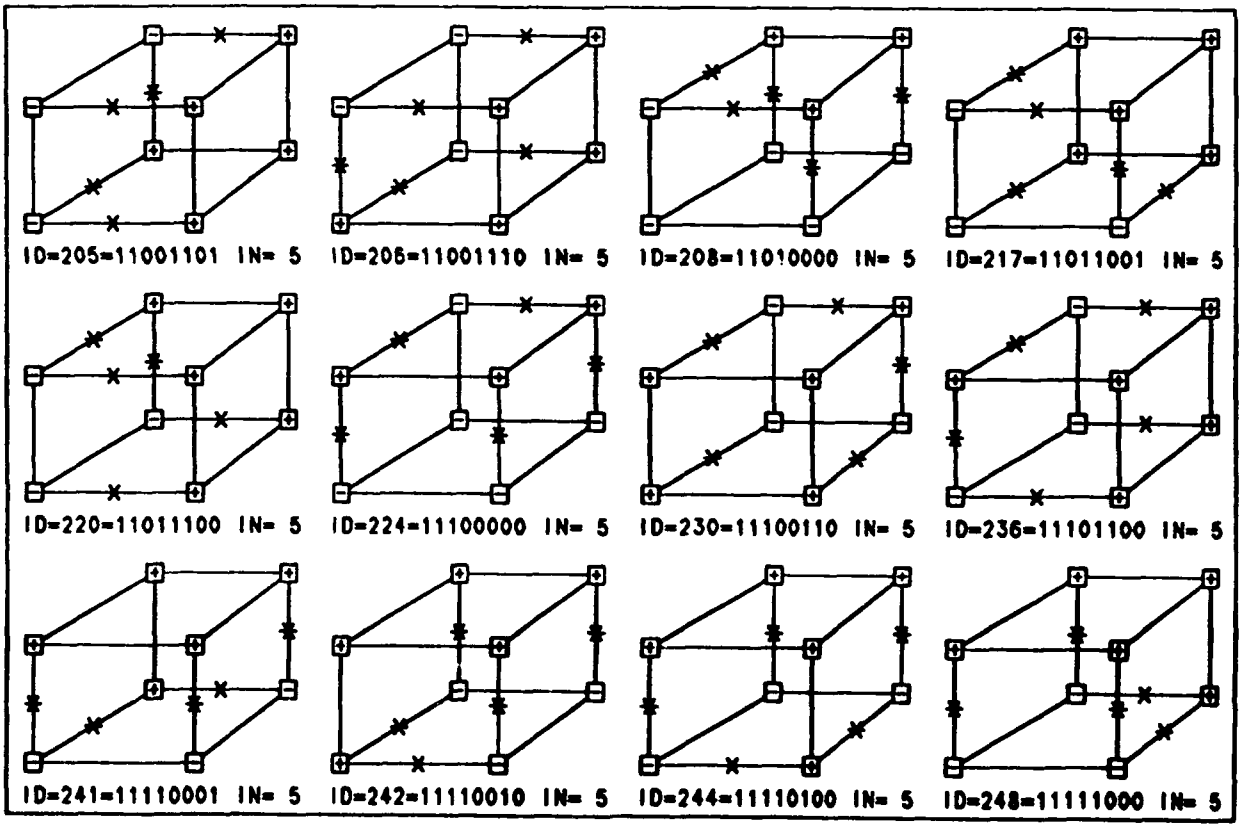

ID=102=01100110 IN= 4
ID=111=01101111 IN= 4
ID=119=01110111 IN= 4
ID=136=10001000 IN= 4
ID=144=10010000 IN= 4
ID=153=10011001 IN= 4
ID=159=10011111 IN= 4
ID=187=10111011 IN= 4
ID=192=11000000 IN= 4
ID=204=11001100 IN= 4
ID=207=11001111 IN= 4
ID=221=11011101 IN= 4

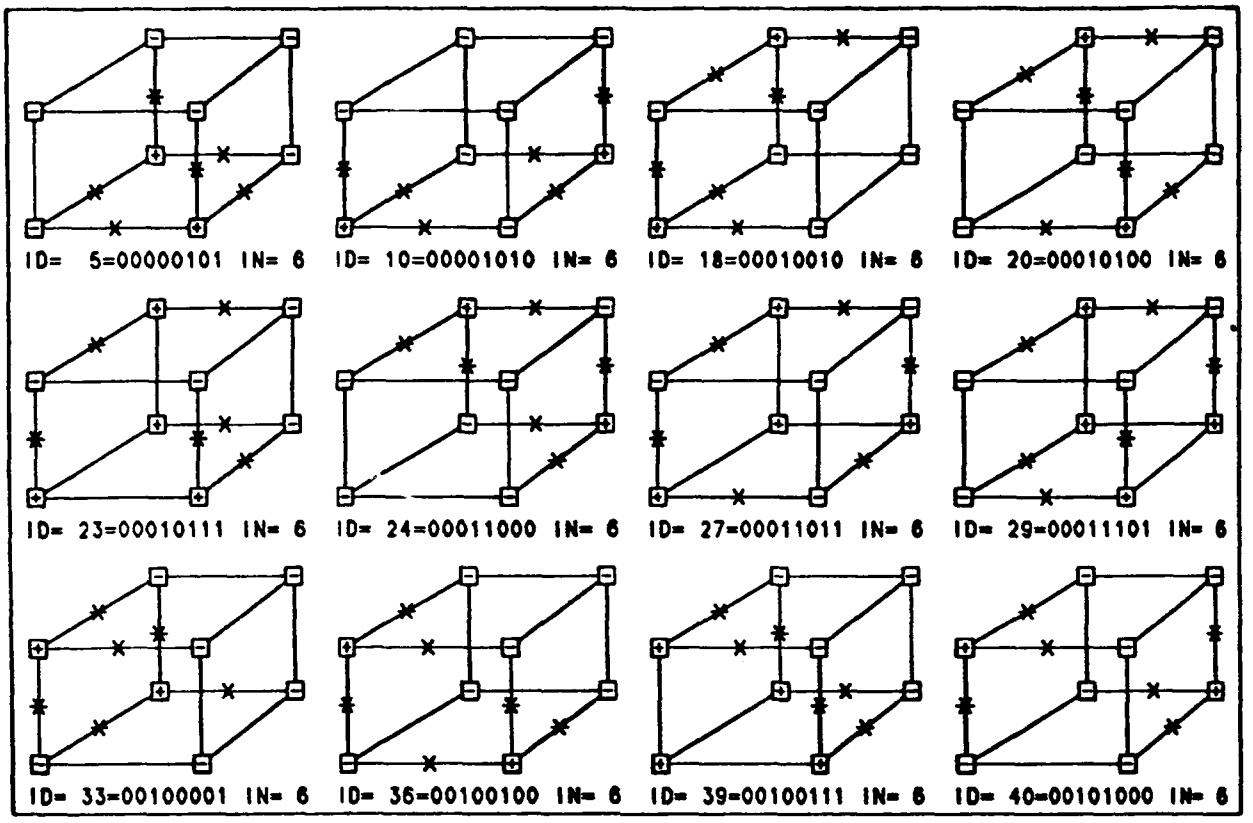

ID=238=11101110 IN= 4
ID=240=11110000 IN= 4
ID=243=11110011 IN= 4
ID=246=11110110 IN= 4
ID=249=11111001 IN= 4
ID=252=11111100 IN= 4

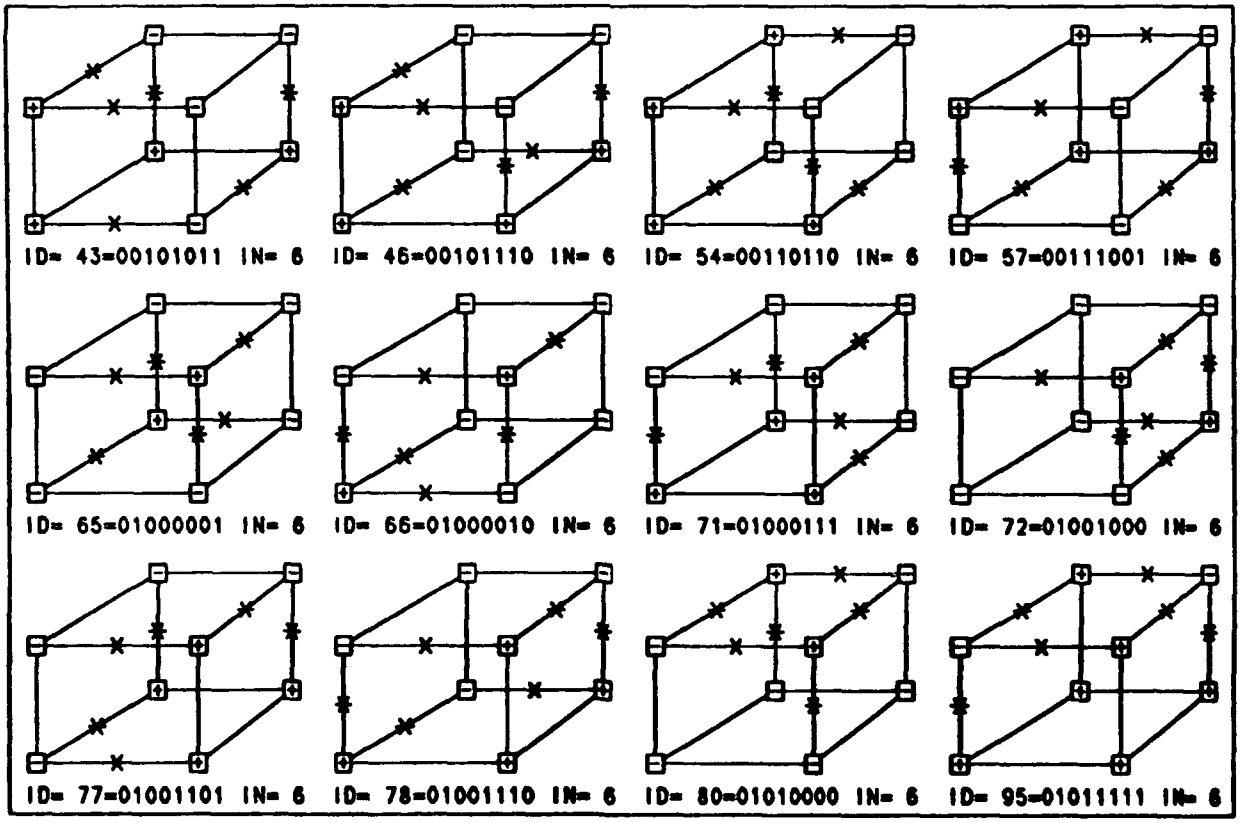

ID= 7=00000111 IN= 5
ID= 11=00001011 IN= 5
ID= 13=00001101 IN= 5
ID= 14=00001110 IN= 5
ID= 19=00010011 IN= 5
ID= 25=00011001 IN= 5
ID= 31=00011111 IN= 5
ID= 35=00100011 IN= 5
ID= 38=00100110 IN= 5
ID= 47=00101111 IN= 5
ID= 49=00110001 IN= 5
ID= 50=00110010 IN= 5

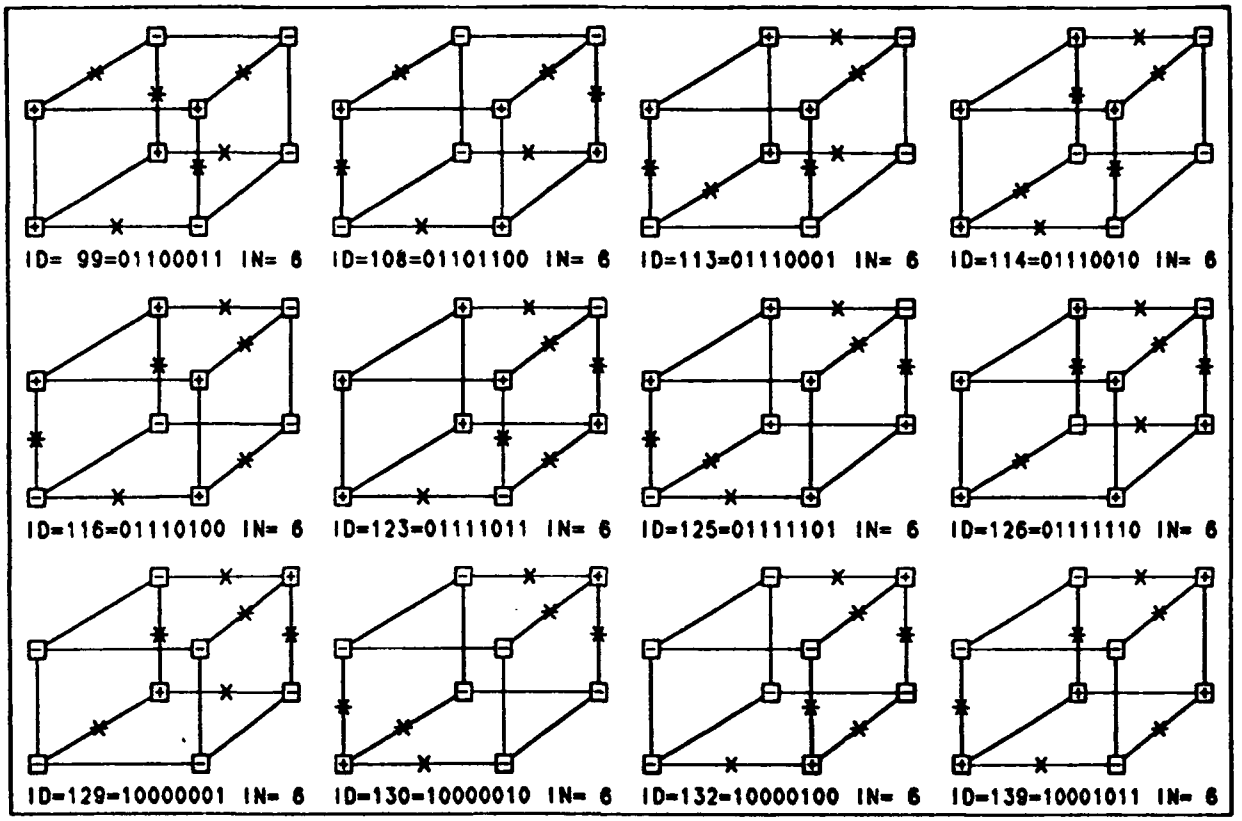

ID= 55=00110111 IN= 5
ID= 59=00111011 IN= 5
ID= 70=01000110 IN= 5
ID= 76=01001100 IN= 5
ID= 79=01001111 IN= 5
ID= 98=01100010 IN= 5
ID=100=01100100 IN= 5
ID=103=01100111 IN= 5
ID=110=01101110 IN= 5
ID=112=01110000 IN= 5
ID=115=01110011 IN= 5
ID=118=01110110 IN= 5

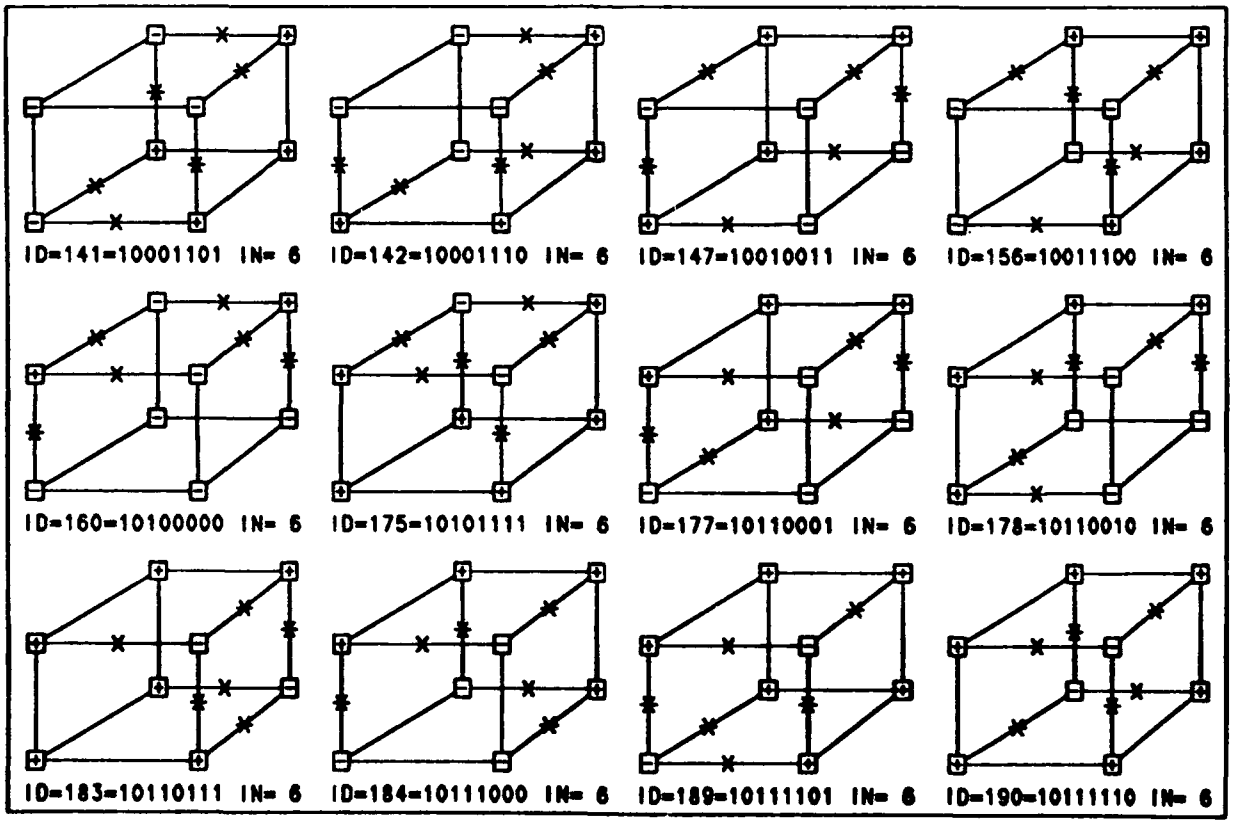

ID=137=10001001 IN= 5
ID=140=10001100 IN= 5
ID=143=10001111 IN= 5
ID=145=10010001 IN= 5
ID=152=10011000 IN= 5
ID=155=10011011 IN= 5
ID=157=10011101 IN= 5
ID=176=10110000 IN= 5
ID=179=10110011 IN= 5
ID=185=10111001 IN= 5
ID=196=11000100 IN= 5
ID=200=11001000 IN= 5

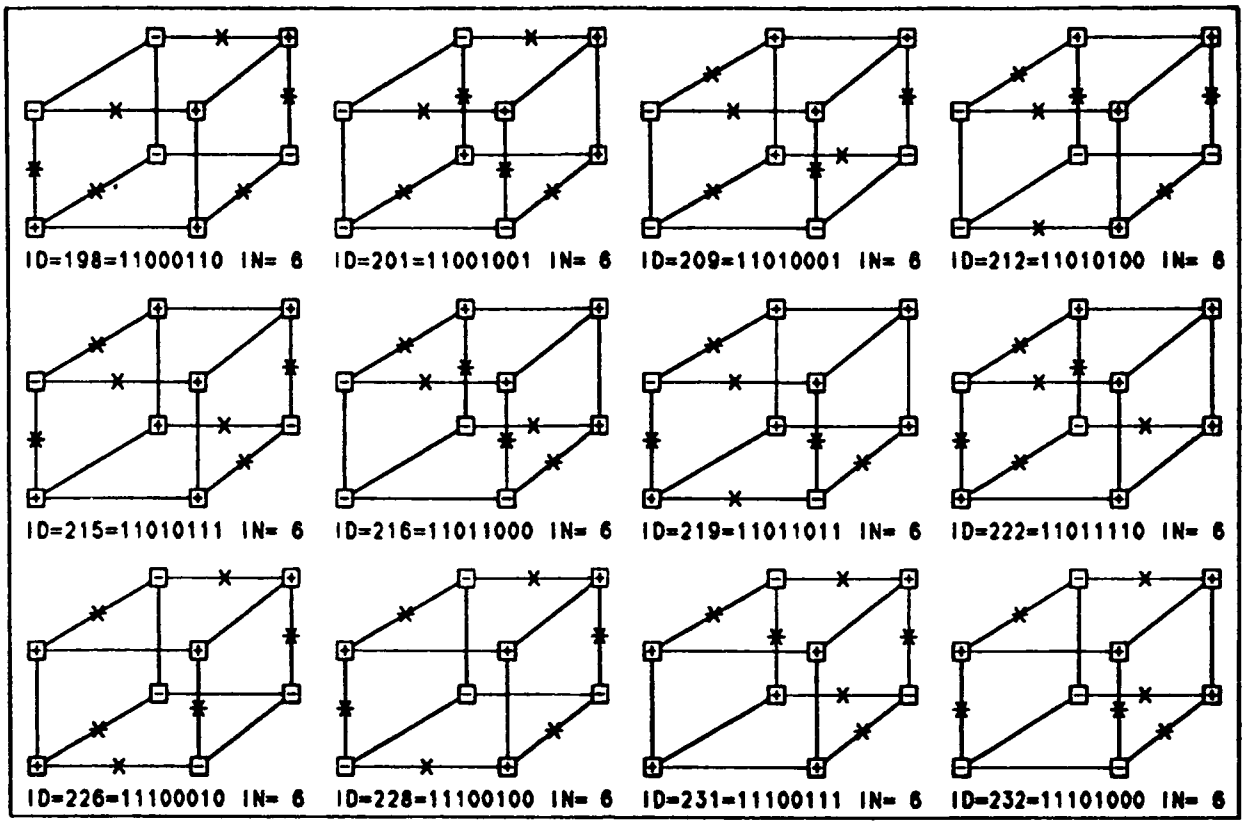

ID=205=11001101 IN= 5
ID=208=11001110 IN= 5
ID=208=11010000 IN= 5
ID=217=11011001 IN= 5
ID=220=11011100 IN= 5
ID=224=11100000 IN= 5
ID=230=11100110 IN= 5
ID=236=11101100 IN= 5
ID=241=11110001 IN= 5
ID=242=11110010 IN= 5
ID=244=11110100 IN= 5
ID=248=11111000 IN= 5

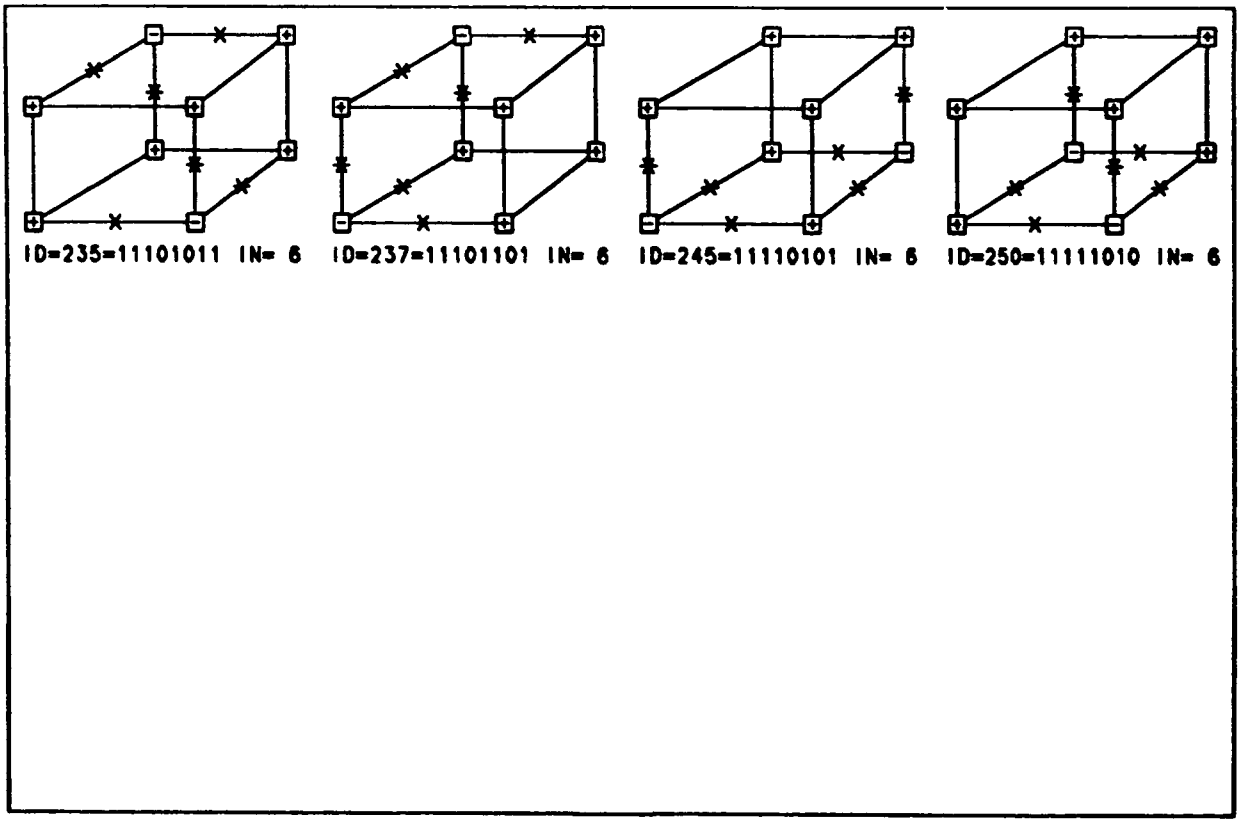

ID= 5=00000101 IN= 6
ID= 10=00001010 IN= 6
ID= 18=00010010 IN= 6
ID= 20=00010100 IN= 6
ID= 23=00010111 IN= 6
ID= 24=00011000 IN= 6
ID= 27=00011011 IN= 6
ID= 29=00011101 IN= 6
ID= 33=00100001 IN= 6
ID= 36=00100100 IN= 6
ID= 39=00100111 IN= 6
ID= 40=00101000 IN= 6

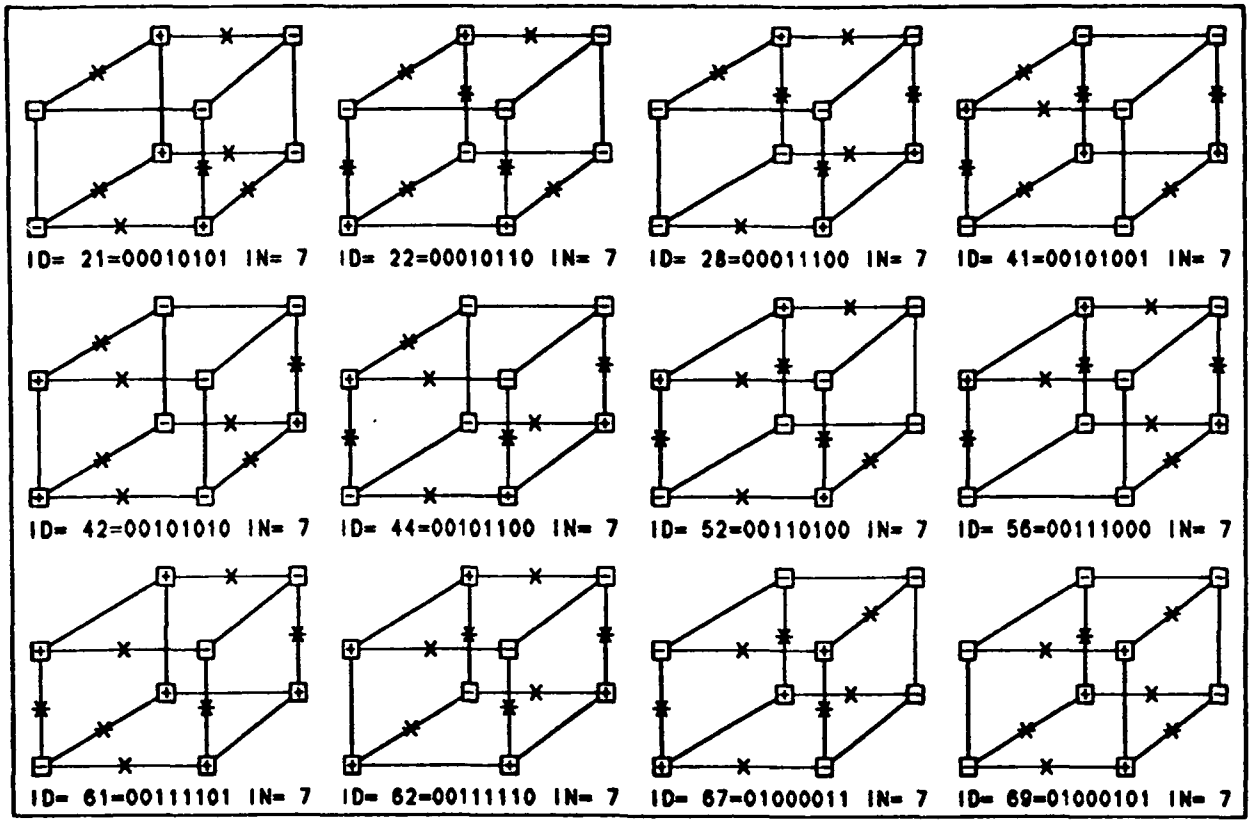

ID= 43=00101011 IN= 6
ID= 46=00101110 IN= 6
ID= 54=00110110 IN= 6
ID= 57=00111001 IN= 6
ID= 65=01000001 IN= 6
ID= 66=01000010 IN= 6
ID= 71=01000111 IN= 6
ID= 72=01001000 IN= 6
ID= 77=01001101 IN= 6
ID= 78=01001110 IN= 6
ID= 80=01010000 IN= 6
ID= 95=01011111 IN= 6

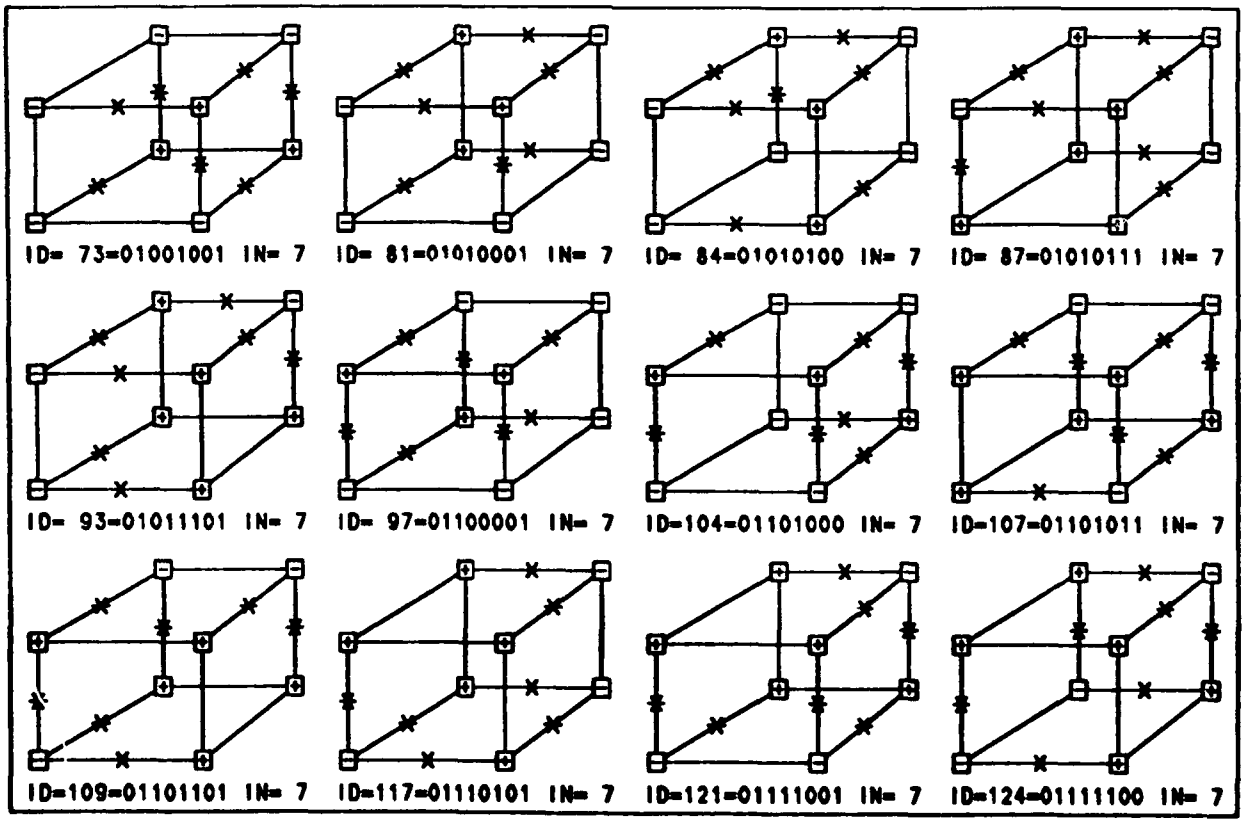

ID= 99=01100011 IN= 6
ID=108=01101100 IN= 6
ID=113=01110001 IN= 6
ID=114=01110010 IN= 6
ID=116=01110100 IN= 6
ID=123=01111011 IN= 6
ID=125=01111101 IN= 6
ID=126=01111110 IN= 6
ID=129=10000001 IN= 6
ID=130=10000010 IN= 6
ID=132=10000100 IN= 6
ID=139=10001011 IN= 6

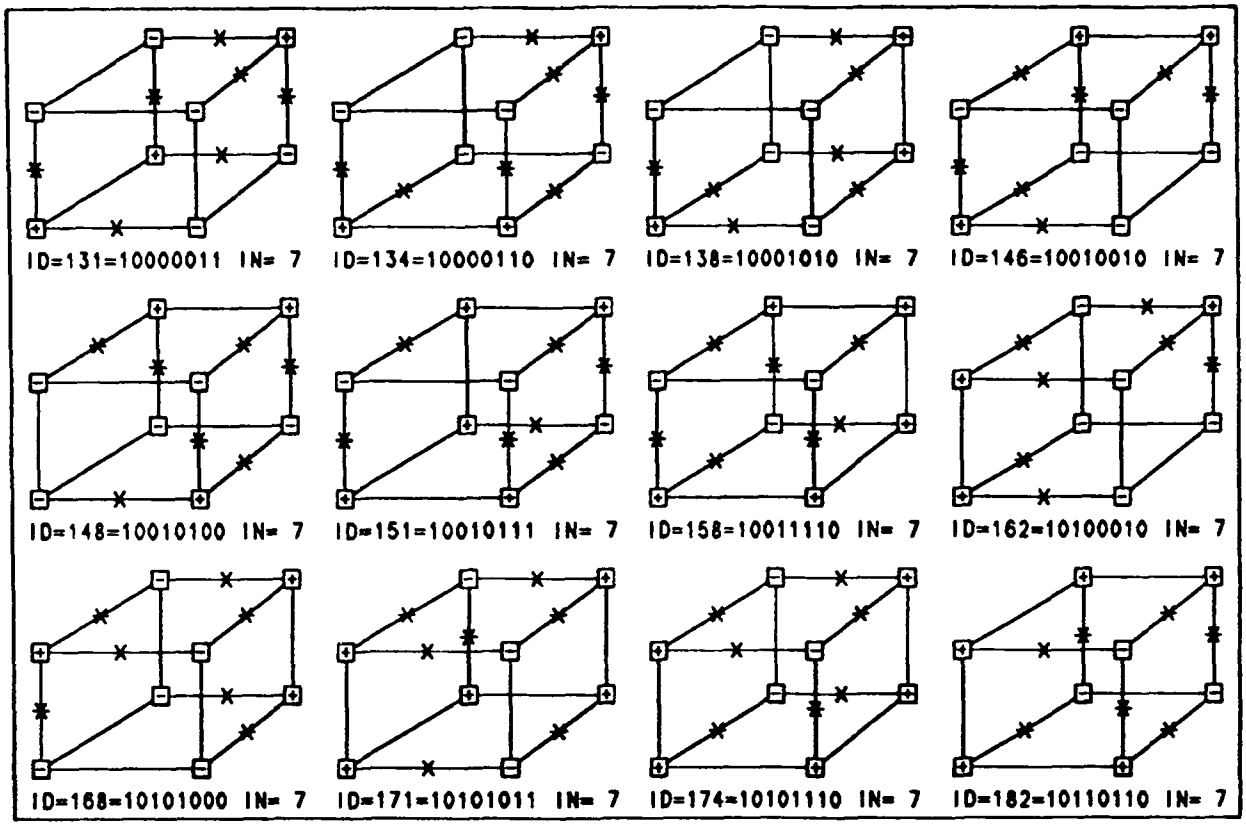

ID=141=10001101 IN= 6
ID=142=10001110 IN= 6
ID=147=10010011 IN= 6
ID=156=10011100 IN= 6
ID=160=10100000 IN= 6
ID=175=10101111 IN= 6
ID=177=10110001 IN= 6
ID=178=10110010 IN= 6
ID=183=10110111 IN= 6
ID=184=10111000 IN= 6
ID=189=10111101 IN= 6
ID=190=10111110 IN= 6

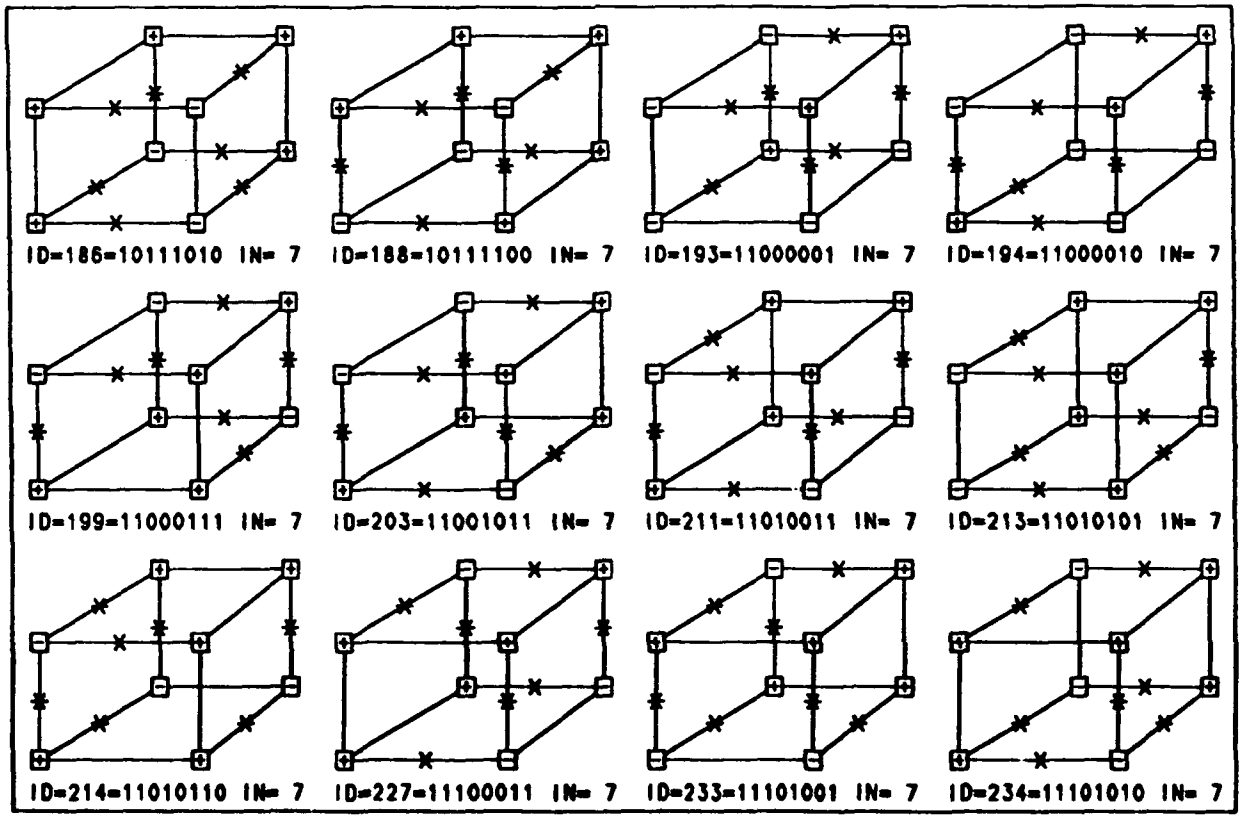

ID=198=11000110 IN= 6
ID=201=11001001 IN= 6
ID=209=11010001 IN= 6
ID=212=11010100 IN= 6
ID=215=11010111 IN= 6
ID=216=11011000 IN= 6
ID=219=11011011 IN= 6
ID=222=11011110 IN= 6
ID=226=11100010 IN= 6
ID=228=11100100 IN= 6
ID=231=11100111 IN= 6
ID=232=11101000 IN= 6

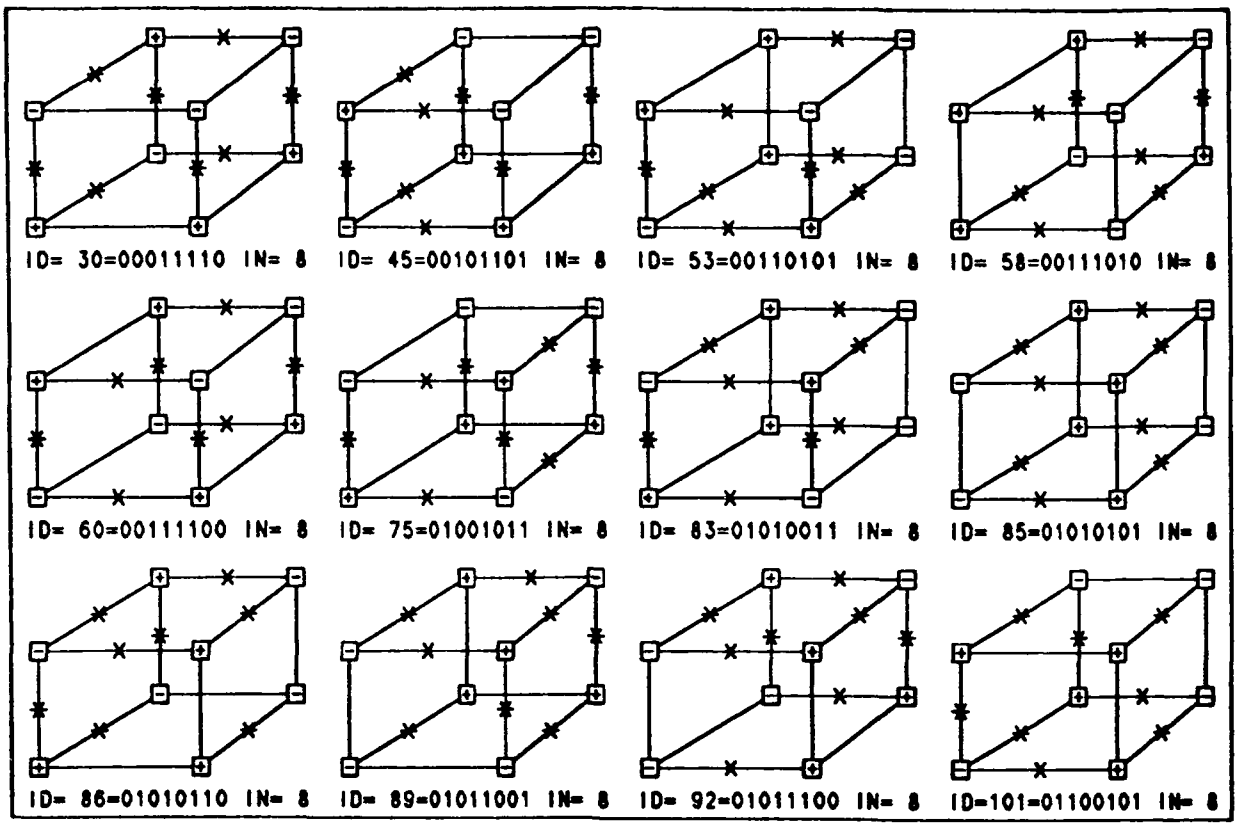

ID=235=11101011 IN= 6
ID=237=11101101 IN= 6
ID=245=11110101 IN= 6
ID=250=11111010 IN= 6

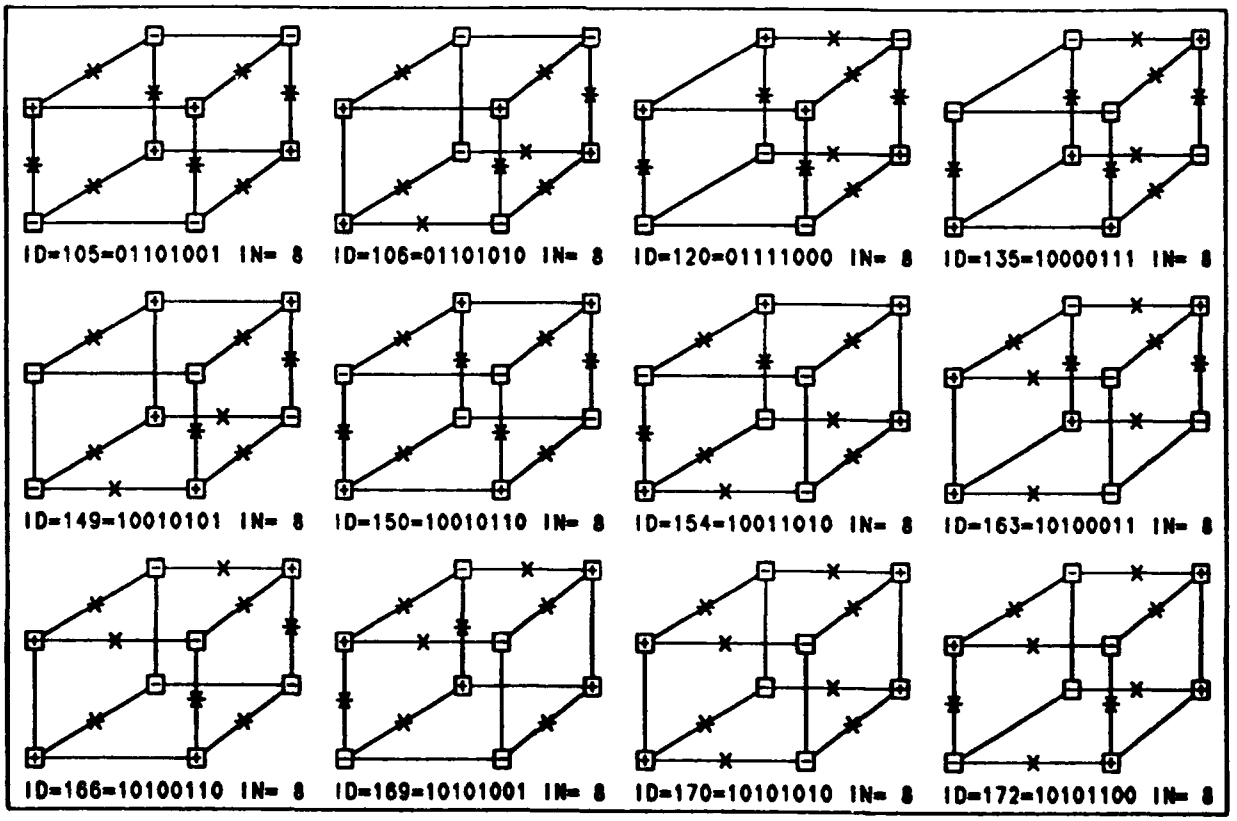

ID= 21=00010101 IN= 7
ID= 22=00010110 IN= 7
ID= 28=00011100 IN= 7
ID= 41=00101001 IN= 7
ID= 42=00101010 IN= 7
ID= 44=00101100 IN= 7
ID= 52=00110100 IN= 7
ID= 56=00111000 IN= 7
ID= 61=00111101 IN= 7
ID= 62=00111110 IN= 7
ID= 67=01000011 IN= 7
ID= 69=01000101 IN= 7

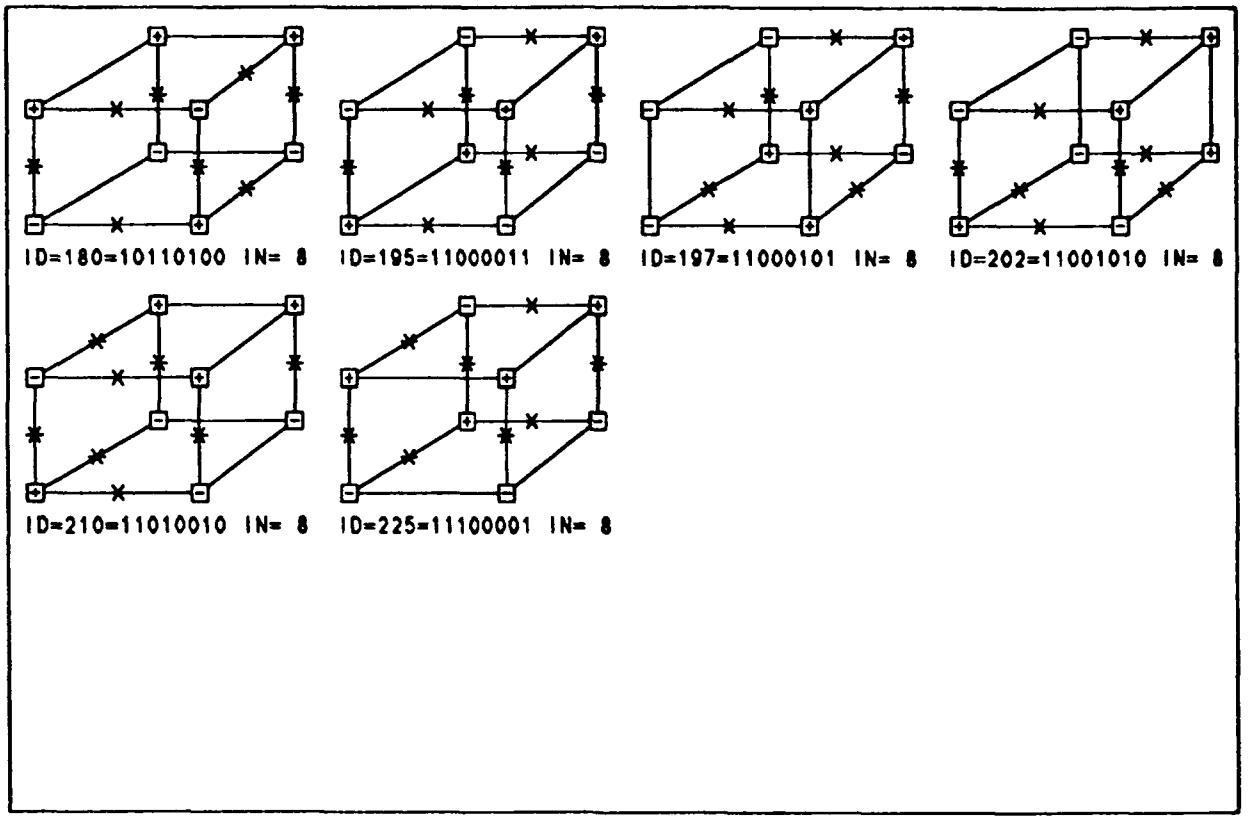

ID= 73=01001001 IN= 7
ID= 81=01010001 IN= 7
ID= 84=01010100 IN= 7
ID= 87=01010111 IN= 7
ID= 93=01011101 IN= 7
ID= 97=01100001 IN= 7
ID=104=01101000 IN= 7
ID=107=01101011 IN= 7
ID=109=01101101 IN= 7
ID=117=01110101 IN= 7
ID=121=01111001 IN= 7
ID=124=01111100 IN= 7

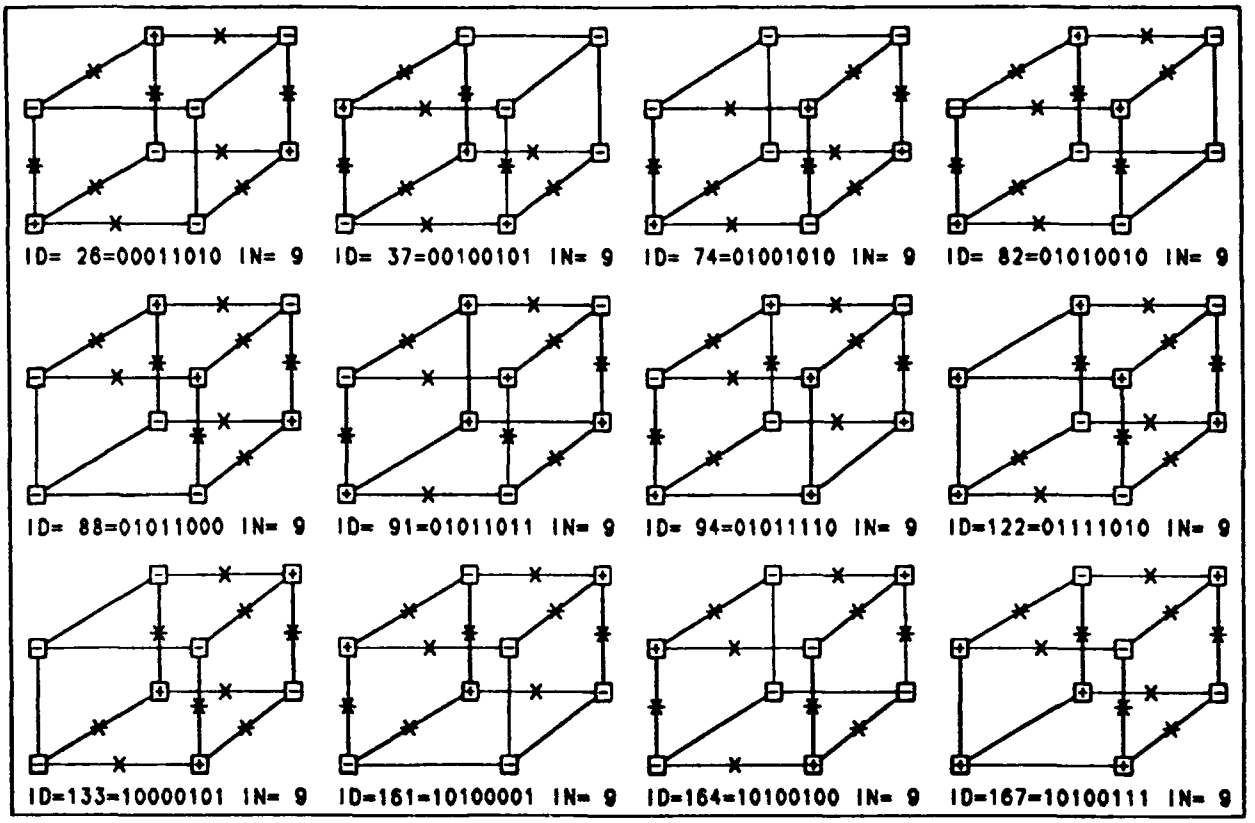

ID=131=10000011 IN= 7
ID=134=10000110 IN= 7
ID=138=10001010 IN= 7
ID=146=10010010 IN= 7
ID=148=10010100 IN= 7
ID=151=10010111 IN= 7
ID=158=10011110 IN= 7
ID=162=10100010 IN= 7
ID=168=10101000 IN= 7
ID=171=10101011 IN= 7
ID=174=10101110 IN= 7
ID=182=10110110 IN= 7

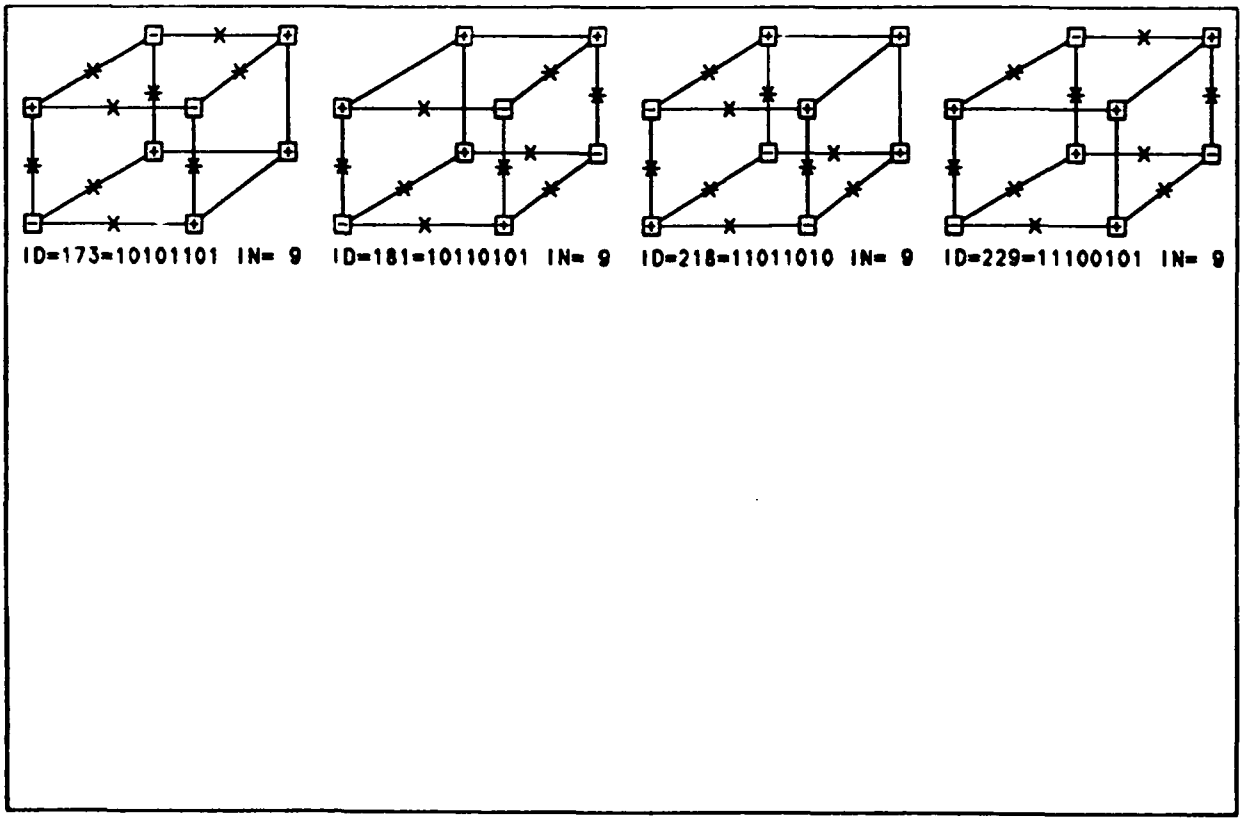

ID=186=10111010 IN= 7
ID=188=10111100 IN= 7
ID=193=11000001 IN= 7
ID=194=11000010 IN= 7
ID=199=11000111 IN= 7
ID=203=11001011 IN= 7
ID=211=11010011 IN= 7
ID=213=11010101 IN= 7
ID=214=11010110 IN= 7
ID=227=11100011 IN= 7
ID=233=11101001 IN= 7
ID=234=11101010 IN= 7

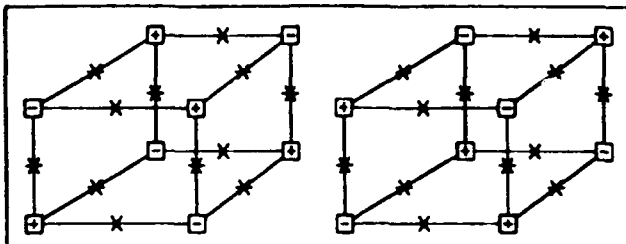

ID= 30=00011110 IN= 8
ID= 45=00101101 IN= 8
ID= 53=00110101 IN= 8
ID= 58=00111010 IN= 8
ID= 60=00111100 IN= 8
ID= 75=01001011 IN= 8
ID= 83=01010011 IN= 8
ID= 85=01010101 IN= 8
ID= 86=01010110 IN= 8
ID= 89=01011001 IN= 8
ID= 92=01011100 IN= 8
ID=101=01100101 IN= 8


ID=105=01101001 IN= 8
ID=106=01101010 IN= 8
ID=120=01111000 IN= 8
ID=135=10000111 IN= 8
ID=149=10010101 IN= 8
ID=150=10010110 IN= 8
ID=154=10011010 IN= 8
ID=163=10100011 IN= 8
ID=166=10100110 IN= 8
ID=169=10101001 IN= 8
ID=170=10101010 IN= 8
ID=172=10101100 IN= 8


ID=180=10110100 IN= 8
ID=195=11000011 IN= 8
ID=197=11000101 IN= 8
ID=202=11001010 IN= 8
ID=210=11010010 IN= 8
ID=225=11100001 IN= 8


ID= 26=00011010 IN= 9
ID= 37=00100101 IN= 9
ID= 74=01001010 IN= 9
ID= 82=01010010 IN= 9
ID= 88=01011000 IN= 9
ID= 91=01011011 IN= 9
ID= 94=01011110 IN= 9
ID=122=01111010 IN= 9
ID=133=10000101 IN= 9
ID=161=10100001 IN= 9
ID=164=10100100 IN= 9
ID=167=10100111 IN= 9


ID=173=10101101 IN= 9
ID=181=10110101 IN= 9
ID=218=11011010 IN= 9
ID=229=11100101 IN= 9



ID= 90=01011010 IN=12 ID=165=10100101 IN=12

Appendix B

Hierarchical Data Structure

Several different data structures are being used to represent topological neighboring relationship in the boundary representation of Solid modeling systems. To supply the adjacency information within a box, the simple Hierarchical data structure is used in the algorithm. For a given edge, as an example, the algorithm requires the information of two faces which share the edge, and two vertices which are limiting the edge.

The Hierarchical data structure treats all the faces of the polyhedra as roots. Furthermore, each face has information of surrounding edges. Each edge has information of two bounding vertices as its children. The structure is composed of 2 tables, Face-Edge Table (FET) and Edge-Vertex Table (EVT). Figure B.1 shows those relations, where following definitions are used.

- F : face
- E : edge
- V : vertex
- nf : total no. of faces

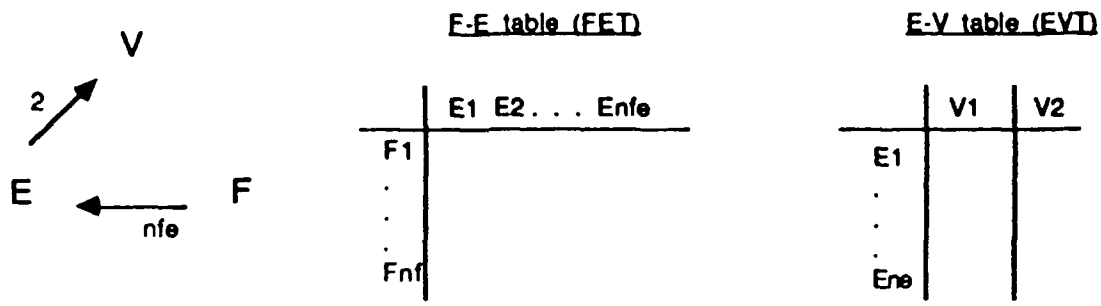


Figure B.1: Relationships in the Hierarchical Data Structure

- ne : total no. of edges
- nv : total no. of vertices
- nee : no. of adjacent edges, given E
- nvv : no. of adjacent vertices, given V
- nve : no. of adjacent edges, given V
- nvf : no. of adjacent faces, given V
- nfe : no. of adjacent edges, given F
- nfv : no. of adjacent vertices, given F
- nff : no. of adjacent faces, given F

Generally, the data structure will be used to answer the following 9 topological queries ;

- T1 : Given V, find all nvv vertices around V
- T2 : Given V, find all nve edges around V
- T3 : Given V, find all nvf faces around V
- T4 : Given E, find 2 vertices connected by E
- T5 : Given E, find all nee edges around E
- T6 : Given E, find 2 faces that meet at E
- T7 : Given F, find all nfv vertices around F
- T8 : Given F, find all nfe edges around F
- T9 : Given F, find all nff faces around F

For a box, the Hierarchical data structure has a fixed form. If we use the local identification system of Figure 3.2, the two tables have the form as Table B.1 and Table B.2.

Another table is created to get the grid position from the base vertex which is the grid point with the lowest (i, j, k) tuple within the box. The table has the differences of grid positions between each vertex and the base vertex. Table B.3 shows the relation.

	E_1	E_2	E_3	E_4
F_1	1	2	3	4
F_2	1	9	5	10
F_3	2	11	6	10
F_4	3	12	7	11
F_5	4	9	8	12
F_6	5	6	7	8

Table B.1: Face-Edge Table

	V_1	V_2
E_1	1	2
E_2	2	3
E_3	3	4
E_4	4	1
E_5	5	6
E_6	6	7
E_7	7	8
E_8	8	5
E_9	1	5
E_{10}	2	6
E_{11}	3	7
E_{12}	4	8

Table B.2: Edge-Vertex Table

	Δi	Δj	Δk
V_1	0	0	0
V_2	1	0	0
V_3	1	1	0
V_4	0	1	0
V_5	0	0	1
V_6	1	0	1
V_7	1	1	1
V_8	0	1	1

Table B.3: Vertex Difference Table